UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA

"Prototipo de Software de Reconocimiento Óptico del Alfabeto Internacional del Lenguaje de Sordos"

TRABAJO DE GRADUACIÓN

Para obtener al grado de:

Ingeniero en Electrónica

Presentado por:

Joaquín Armando Fonseca Rosales

Mario Ernesto Orellana Crespín

Juan Carlos Rivera García

Asesor:

Ing. Sergio Adrián Martín

San Salvador, El Salvador, Centro América

Septiembre de 2005.

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

AUTORIDADES:

RECTOR:
ING. FEDERICO MIGUEL HUGUET RIVERA

VICERRECTOR ACADÉMICO:
PBRO. VÍCTOR BERMÚDEZ YANEZ, SDB

SECRETARIO GENERAL:
LIC. MARIO RAFAEL OLMOS ARGUETA

DECANO DE LA FACULTAD DE INGENIERÍA:
ING. ERNESTO GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE ELECTRÓNICA:
ING. OSCAR GIOVANNI DURÁN VIZCARRA

ASESOR DEL TRABAJO DE GRADUACIÓN:
ING. SERGIO ADRIÁN MARTÍN

JURADO EVALUADOR:
ING. EDUARDO RIVERA
ING. ANA DAYSI MONTECINO
DR. JORGE ERNESTO LEMUS SANDOVAL

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN

Ing. Eduardo Rivera
JURADO

Ing. Ana Daysi Montecino
JURADO

Dr. Jorge Lemus Sandoval
JURADO

Ing. Sergio Adrián Martín
ASESOR

AGRADECIMIENTOS

Quiero agradecer a Dios por darme la oportunidad de haber culminado la elaboración de este trabajo y también por haber derramado bendiciones sobre mi persona.

A la vez agradecer a mis padres, Joaquín Armando Fonseca y Ana Concepción de Fonseca, por darme todo su apoyo, comprensión y sacrificio en todo el tiempo que duro la realización del trabajo.

También agradecer a mis hermanas, Maritza Fonseca, Carolina Fonseca y Guadalupe Fonseca que me apoyaron y estuvieron conmigo en los momentos difíciles, ya que siempre permanecieron pendientes y me dieron ese empuje para salir adelante.

A mi esposa, Fara Galeano y mis hijos, Armando Fonseca y Fernando Fonseca, por estar siempre conmigo, por darme su amor y su apoyo, por todos aquellos desvelos que se dieron en el transcurso del proceso.

También agradecer a todas aquellas personas que con sus oraciones pedían por que culminara con éxito el trabajo de graduación, por que siempre permanecieron pendientes y me brindaron todo el apoyo para seguir adelante en el desarrollo de este trabajo.

Joaquín Armando Fonseca Rosales

AGRADECIMIENTOS

Gracias a Dios sobre todas las cosas, por darme salud, sabiduría, paciencia y fuerza para lograr finalizar este proyecto.

Agradezco a mis padres, ya que sin ellos, sin su apoyo y su gran fe en mí no hubiese logrado vencer todas las dificultades que me ha presentado la vida. A mi papá, por ser mi modelo de vida y el hombre que más admiro, porque siempre me ha enseñado con el ejemplo valores como la responsabilidad, honestidad y amor al prójimo. Gracias a mi mamá por el soporte incondicional, por confortarme y siempre anticiparse a mis necesidades.

Agradezco a mi familia: Wendy, Fernando y Mario porque han sido los más sacrificados en este tiempo de trabajo. Gracias por haber sido mi soporte en los momentos difíciles, por su paciencia y por todo su amor. Es solamente por ellos que tiene sentido todo mi esfuerzo.

Gracias Jorge, por tu ayuda sustancial, pero sobre todo por ser un verdadero amigo.

Gracias Armando y Juan, porque más allá del trabajo está la amistad que me han brindado.

Agradezco a todos mis amigos, por sus oraciones, por darme ánimos y su confianza, como también por las palabras de aliento cuando las necesitaba.

Gracias a todos aquellos familiares y amigos que no menciono y han sido parte de mi vida durante este trabajo, para quienes también ofrezco mis agradecimientos.

Mario E. Orellana Crespín.

AGRADECIMIENTOS

En primer lugar quiero dar gracias a Dios por permitirme llegar a esta etapa en mi vida, por estar siempre presente con sus bendiciones, darme la confianza y fuerza necesaria para lograr finalizar satisfactoriamente este trabajo.

Quiero dar gracias a mi madre, Ana María García, que constantemente ha mantenido sus oraciones y me ha apoyado en toda mi vida. A mi padre, German Rivera Rodriguez, que ha estado presente en el momento que lo he necesitado.

Gracias a Diana Gómez Canjura por brindarme todo su apoyo, por estar a mi lado, por compartir mis logros y tristezas, por darme ánimos para seguir adelante ante cualquier circunstancia.

A nuestro asesor el Ing. Sergio Martín, por prestar sus conocimientos, su tiempo y mantenerse pendiente de todos los pasos administrativos a realizar para finalizar este trabajo.

A mis familiares que se preocuparon y me apoyaron de diversas formas, y mantuvieron comunicación mostrando interés en mi proceso de graduación.

A los familiares de mis compañeros Mario Orellana y Joaquín Fonseca, por todas las atenciones que mostraron a mi persona y por ayudarnos en lo que necesitábamos.

A mis amigos y compañeros que siempre me han dado ánimos en momentos difíciles han confiado en mí y me han apoyado desde el inicio de mi carrera.

A todas las personas que nos ayudaron en este trabajo, dándonos consejos e ideas para corregir errores y optimizarlo, a los que se prestaron para tomarles fotos a sus manos y poder utilizarlas para los entrenamientos de las redes neuronales.

Juan Carlos Rivera García

CONTENIDOS

INTRODUCCIÓN

El trabajo que a continuación se expone, trata sobre la creación de una herramienta de software que sea capaz de realizar diversos procesos orientados al reconocimiento de las señas del lenguaje de sordos, correspondientes al alfabeto internacional. Para lograr tal objetivo, se plantea el uso de diferentes tareas de procesamiento digital de imágenes, tales como, captura de imagen con webcam, conversión a escala de grises, detección de bordes, reducción o compresión de los datos que se obtienen de la extracción de características propias de cada seña. Para el reconocimiento de la imagen, incluyendo las variaciones posibles que se presentan con cada una de las señas, se plantea el uso de una red neuronal, concentrándose en el desarrollo de la tipo Retro-propagación, presentando las dificultades y ventajas encontradas; como otra opción, el uso de valores propios de la imagen “eigvalues”.

Además, este trabajo pretende ser una referencia para el futuro desarrollo de aplicaciones más completas, por lo que se presentan algunas pruebas realizadas con sus respectivos resultados con el fin de que esta información sirva como precedente al lector que se interesa en conocer los aspectos básicos que le permitan escoger un tipo de red para desarrollar una aplicación de procesamiento y reconocimiento de imágenes.

Debido a que se implementará un prototipo de software que realice la captura de imágenes de manos representando las señas del alfabeto internacional del lenguaje de sordos, es necesario utilizar equipo de iluminación, dispositivo de captura y computadoras en las que se realizará el procesamiento de la seña.

Los dispositivos necesarios son fácilmente encontrados en el mercado, para la iluminación se pueden utilizar lámparas fluorescentes de luz blanca, halógenos, etc. En cuanto al dispositivo de captura, se requiere una webcam que tenga como mínimo resoluciones de 120x160 píxeles, además se necesita una computadora con un

procesador de alta velocidad y memoria suficiente para los diversos procesos que llevan al reconocimiento de la seña presentada.

Este prototipo será desarrollado para sistema operativo Windows 98 en adelante, por ser el más utilizado y más conocido por las personas, a la vez tiene mayor compatibilidad con programas tales como matlab, visual basic, visual c++, etc.

El proyecto está destinado a ser utilizado en aulas iluminadas donde se impartan clases a personas que utilicen el lenguaje de señas para comunicarse.
Como el software es un prototipo que se basa en el reconocimiento óptico de patrones, éste puede ser tomado como base para otras aplicaciones similares como reconocimiento de rostros, firmas, huellas digitales, sistemas de seguridad, etc.

Prototipo de Software de Reconocimiento Óptico del Alfabeto Internacional del Lenguaje de Sordos

1 PROCESAMIENTO DE IMÁGENES

1.1 Las imágenes digitales en dos dimensiones.

Las imágenes digitales[1] en dos dimensiones se dividen en dos tipos: imágenes vectoriales y de mapa de bits. Esta no es una división tajante, ya que las imágenes vectoriales suelen admitir la incrustación de imágenes de mapa de bits en su interior.

Las imágenes de mapa de bits se pueden representar, ver Figura 1, mediante retículas de celdillas a las que se van asignando valores. Este modo es la base de todas las imágenes impresas y de buena parte de las digitales.

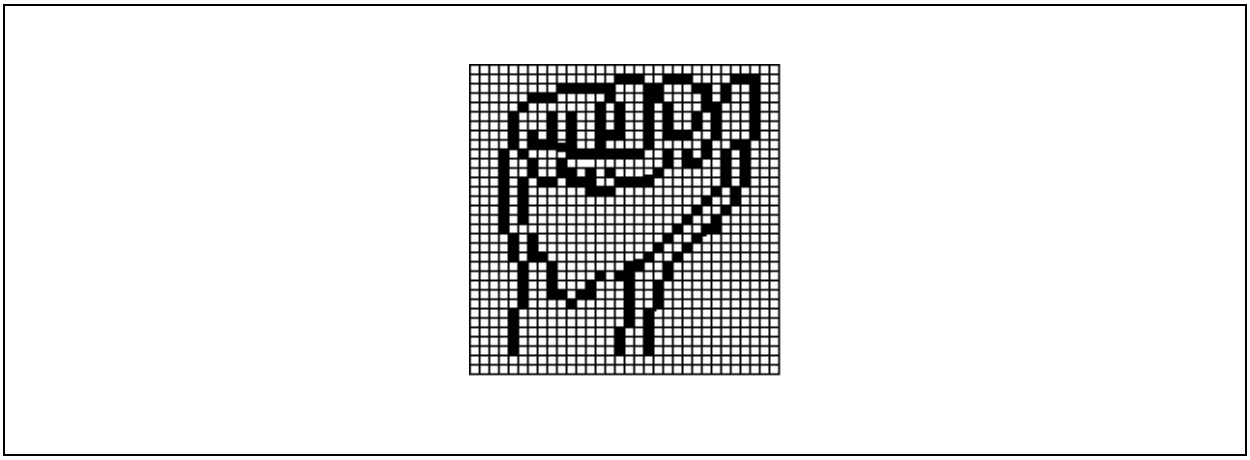


Figura 1- Imagen de mapa de bits, cada celda tiene asignado un valor determinado.

Las imágenes digitales en dos dimensiones se realizan creando una retícula de cuatro lados, iguales de dos a dos (ancho y alto). Cada una de las celdillas de dicha retícula se llama "píxel". Un píxel, es un concepto inmaterial que no tiene una medida concreta. No

[1] Refiérase a ésta dirección:http://www.andygoldstein.org/lapaginadelalumno/materias/contenidos/colordigital/

se puede decir si un píxel mide 1 cm. o 1 Km. En principio, es solamente una medida de división en celdillas.

De este modo, se puede hablar de una imagen que tenga 200 × 100 píxeles sin saber que tamaño real y físico tiene. Lo único que se sabe es que se ha dividido en 20,000 celdillas.

Sin embargo, cuando se le asigna a esa imagen una resolución, entonces sí se conocerá su tamaño. Por ejemplo, si se dice que tiene 100 píxeles por pulgada, se refiere que cada 2.54 cm. (eso es lo que mide una pulgada), habrá 100 celdillas, con lo que cada píxel equivaldrá a 0.254 mm. Si se dijera que esa imagen tiene una resolución de 1 píxel por pulgada, entonces ahora esa celdilla tomaría el valor de 2.54 cm.

Todo ello significa, que el píxel es sólo una unidad de división sin un tamaño real concreto. Sólo cuando se asigna una resolución a la imagen de la que se habla, se le proporciona un tamaño concreto al píxel.

Hay imágenes de mayor resolución e imágenes de más baja resolución. A mayor resolución, mayor nitidez del dibujo y mejor se reflejan los detalles. Sin embargo, hay que tener presente que cualquier resolución que supere la que el dispositivo de salida (pantalla, impresora, etc.) es capaz de representar no hace más que sobrecargar el sistema y ralentizar el trabajo.

1.1.1 Tipos de imágenes de mapa de bits.

Una forma muy importante de clasificar las imágenes de mapa de bits es según la cantidad y tipo de información que se asigne a cada píxel, a continuación se presentan tres diferentes tipos de imágenes de mapas de bits:

- **Imágenes de 1 bit por píxel.**

En este tipo de imágenes cada celdilla (píxel) sólo puede tener uno de dos valores: Uno o cero. En la Figura 2 se observan imágenes de 1 bit por píxel, basta 1 bit para definir esa alternativa, se les llama "imágenes de 1 bit" (también se les llama "imágenes de mapa de bits, de alto contraste, o imágenes de línea").

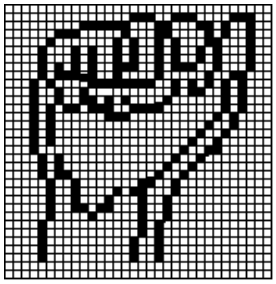
	
(a) Imagen de 32 × 32 píxeles (1024).	**(b) Imagen de 200 × 200 píxeles (40,000).**

Figura 2 - Imágenes de 1 bit por píxel.

- **Imágenes de escala de grises (8 bits por píxel).**

Cada píxel puede tener 256 valores diferentes (las 256 posibilidades combinatorias de un byte u octeto). Este es el modo de las imágenes digitales de blanco y negro "normales". Aunque pueda parecer increíble, en ellas sólo se distinguen hasta 256 tonos diferentes de gris (y no suelen aparecer todos a la vez).

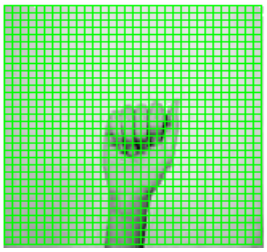
	
(a) Imagen de 32 × 32 píxeles (1024) con 1 byte (8 bits) por píxel.	**(b) 120 × 160 píxeles (19,200) en escala de grises.**

Figura 3 - Imagen en escala de grises.

- **Imágenes RGB o Lab (24 bits por píxel).**

Si se toma un píxel y se le asignan tres bytes, se disponen de 24 bits en tres grupos de ocho, siguiendo el sistema de color de los monitores de televisión, que se basan en tres "canales" de luz de color (Rojo, Azul y Verde). De este modo se pueden distinguir hasta 16,777,216 millones de tonos de color (256 Rojo × 256 Azul × 256 Verde). En realidad, lo que se hace es superponer tres canales de luz, uno rojo, otro verde y otro azul, cada uno con 256 posibilidades de tono. En la Figura 4 se muestra un ejemplo de una imagen RGB.

Figura 4 - Imagen de 120 × 160 píxeles en modo RGB.

1.1.2 El proceso de digitalización (muestreo y cuantificación).

Una imagen natural capturada con una cámara, o cualquier otro tipo de instrumento óptico presenta una variación de sombras y tonos continuos. A este tipo de imágenes se le conocen como imágenes analógicas.

Para que una imagen analógica, en blanco y negro, en escala de grises (las llamadas comúnmente, imágenes en blanco y negro), o a color, pueda ser "manipulada" usando un ordenador, primero deben convertirse a un formato adecuado. Este formato es la imagen digital correspondiente. En la Figura 5 se ejemplifican las imágenes analógicas a color y las imágenes analógicas en escala de grises, estas últimas conocidas como imágenes en blanco y negro.

	
(a) Imagen "analógica" en color.	**(b) Imagen "analógica" en escala de grises.**

Figura 5 - Imágenes analógicas.

La transformación de una imagen analógica a otra discreta se llama digitalización y es el primer paso en cualquier aplicación de procesamiento de imágenes digitales. El proceso de digitalización consta de dos partes: muestreo y cuantificación.

Un muestreo consiste en una subdivisión de la imagen analógica en porciones. En la Figura 6 se observan particiones que envuelven polígonos regulares, esto es, polígonos con lados y ángulos de valor constante en todos ellos. Se puede demostrar que sólo se pueden usar tres tipos de polígonos regulares: triángulos, cuadrados y hexágonos.

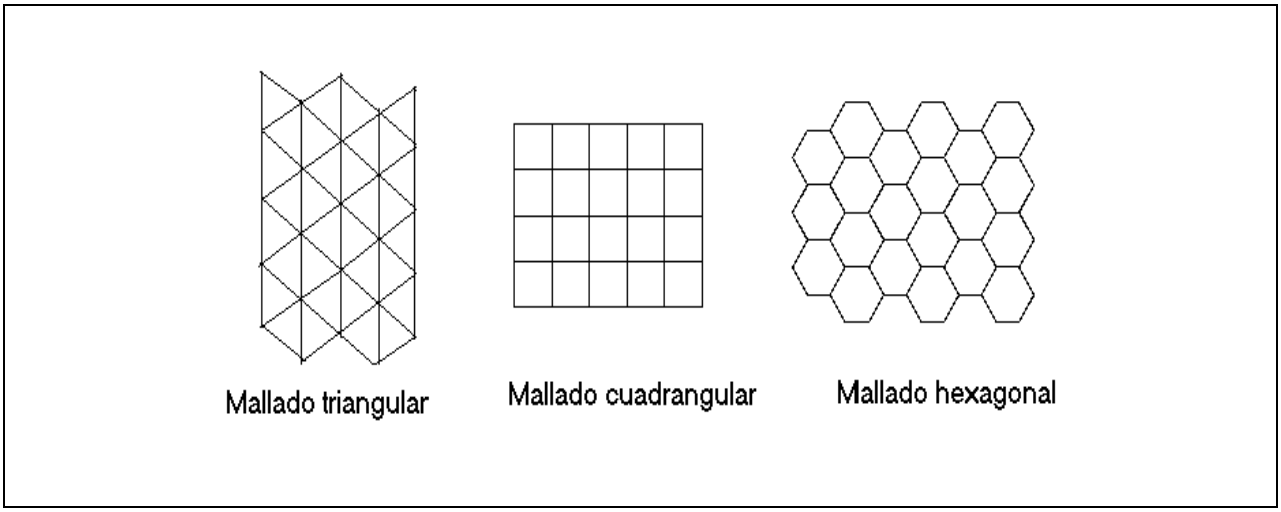


Figura 6 - Tipos de mallados para la digitalización de una imagen.

Cada uno de estos polígonos representan sensores sensibles a la intensidad de luz. La salida de estos sensores es un valor (amplitud) dentro de una escala (color). La salida puede ser, o bien un único valor (escala de grises) o bien un vector con tres valores por polígono (RGB) que se corresponden con la intensidad de color rojo (R), verde (G) y azul (B). La escala de colores también tiene un rango discreto (por ejemplo, de 8 bits = 256 valores). Las imágenes en escala de grises con sólo 2 colores: blanco y negro (0 y 1, respectivamente), se llaman imágenes binarias.

A este proceso de discretización del color se le llama cuantificación y a un polígono de color constante se le llamará píxel.

- **Imágenes en escala de grises.**

La Figura 7 muestra un proceso de digitalización de una imagen.

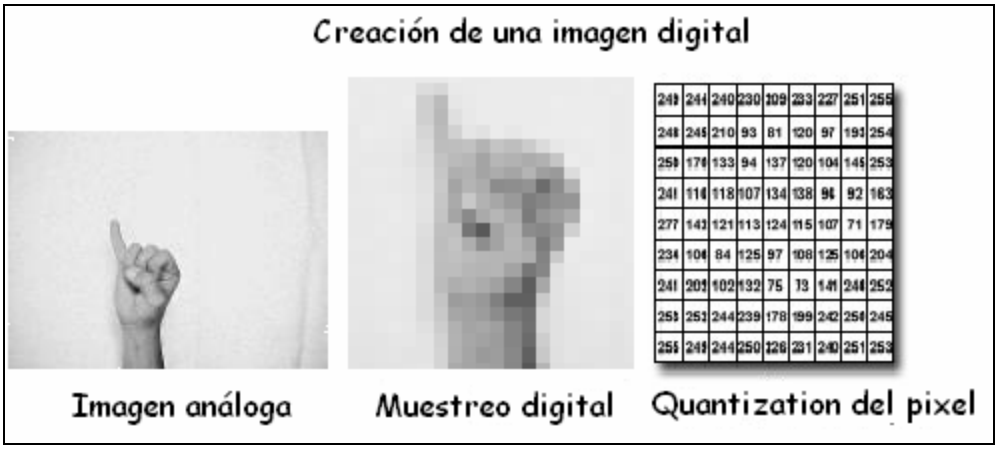


Figura 7 - Pasos a seguir para crear una imagen digital.

En la figura anterior se muestra la creación de una imagen digital, el muestreo se ha hecho usando un mallado cuadrangular de 9 por 9 cuadrados y la cuantificación consiste en una paleta de 256 niveles de gris (donde 0 indica el color negro y 255 el color blanco):

En la Figura 8 se observa que, partiendo de una misma imagen, es muy diferente la imagen digital obtenida, dependiendo del mallado que se escoja:

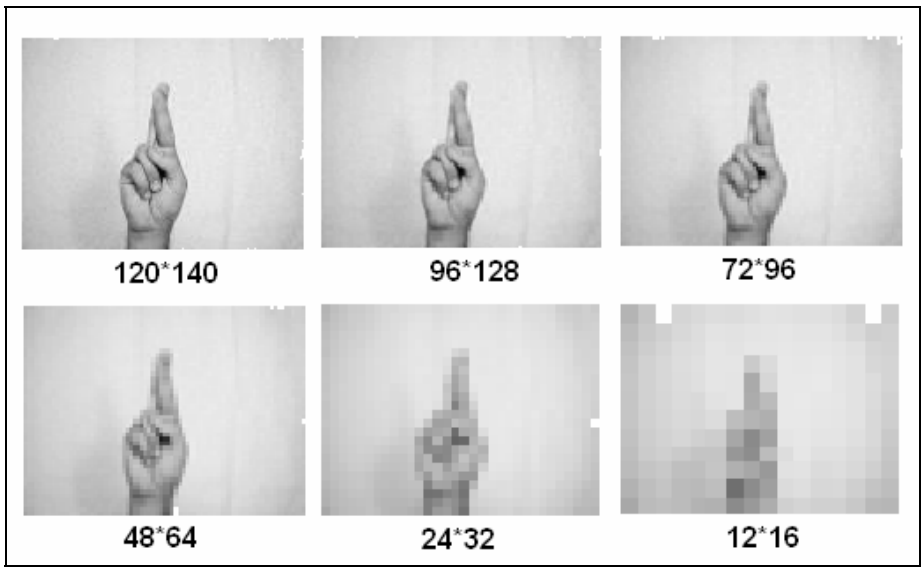


Figura 8 - Efecto espacial de la resolución sobre el pixelado en imágenes digitales.

El efecto espacial de la resolución demuestra que el número de filas y columnas van disminuyendo, el ejemplo se ha tomado desde una de 120*140 hasta una de 12*16, por lo que la calidad de la imagen también disminuye.

En caso de que la resolución aumente, su calidad será notablemente mejor.

También hay que tener en cuenta la paleta de colores, como se observa en el ejemplo siguiente:

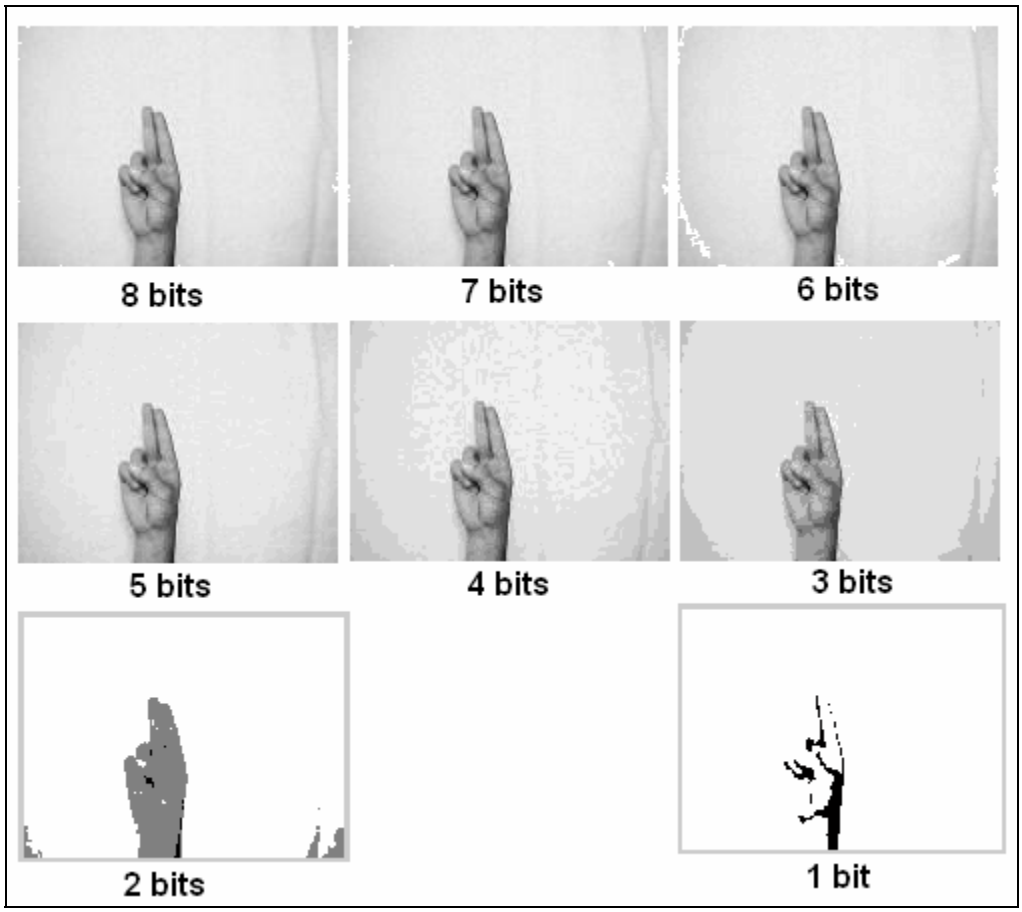


Figura 9 - Resolución en escala de grises y apariencia de la imagen digital.

La resolución (el grado de detalle discernible) de una imagen depende estrechamente de estos dos parámetros (muestreo y cuantificación). Cuanto más se incrementan, más se aproxima la imagen digitalizada a la original, así como se muestra en la Figura 9.

La cantidad de niveles de gris y la finura del mallado que se escojan, deben producir una imagen digital aceptable, en el sentido que no sea perceptible al ojo humano el paso de un color a otro, entre dos píxeles consecutivos.

Sin embargo, se ha de tener en cuenta que si el muestreo consiste en un mallado de N por M cuadrados y el número de niveles de gris permitido son $G = 2^k$,

entonces el número de bits necesarios para almacenar una imagen digitalizada es: N x M x k

Por ejemplo, una imagen de 128 x 128 con 64 niveles de gris necesita 98,304 bits = 96 KB de memoria. Una de 256 x 256 con 132 niveles de gris necesita 458,752 bits = 56 KB. Y una de 1024 x 1024 con 256 niveles de gris necesita 8,388,608 bits = 1024 KB = 1 MB.

En el modelo matemático de una imagen, un píxel se identifica con su centro, pudiendo representar los píxeles como puntos (x,y) del plano, donde (x,y) son las típicas coordenadas cartesianas.

1.2 Detección de Bordes.

La detección de esquinas y líneas se basa en los operadores de detección de bordes[2], éstos, mediante el cálculo de primeras y segundas derivadas permiten determinar puntos de principal importancia para poder realizar las mediciones necesarias.

En el análisis de objetos dentro de las imágenes resulta esencial poder distinguir entre el objeto de interés y el resto de la imagen. Las técnicas utilizadas para determinar los objetos de interés son conocidas como técnicas de segmentación. Una de las más comunes es la segmentación mediante la detección de bordes.

Un borde se puede definir como cualquier discontinuidad que sufre alguna función de intensidad sobre los puntos de la misma, o una frontera entre dos regiones con propiedades de colores o niveles de gris distintas.

La idea básica de las técnicas de detección de bordes es la aplicación de un operador derivativo local para identificar las discontinuidades del nivel de gris.

---

[2] Refiérase a ésta dirección: http://www.us.es/gtocoma/pid/pid8/pid80.htm

Existen diferentes tipos de bordes, estos están basados en los diferentes cambios que se presenten, En la Figura 10 se ilustran y describen los tipos de bordes, estos son los siguientes:

- Cambio brusco en la distancia cámara-objeto (dc).
- Cambio en la normal del objeto (n).
- Cambio en la reflectancia del objeto (r).
- Cambio en la proyección de la luz incidente (s).

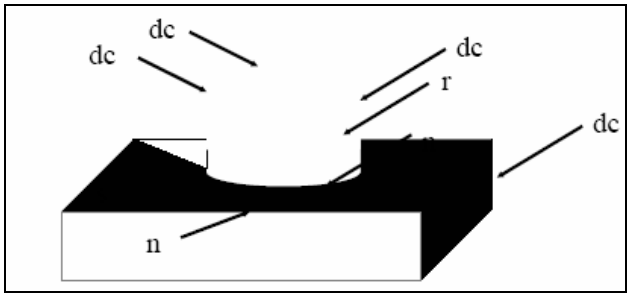


Figura 10 - Tipos de bordes existentes en una imagen.

Existe una gran variedad de métodos para la detección de bordes, mismos que se basan en información con respecto a los límites de una imagen. Los métodos de detección de bordes, utilizan para sus fines, diversos operadores que marcan puntos de acuerdo a discontinuidades en los niveles de gris, los colores o las texturas.

Como ejemplo, la Figura 11 muestra el proceso de la extracción de bordes:

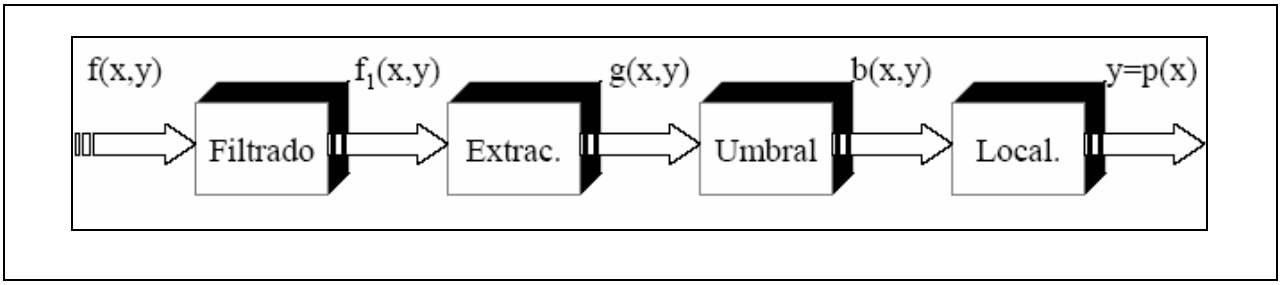


Figura 11 - Procesos requeridos para la extracción de bordes.

Si se tiene una función f(x,y), la detección de bordes consiste en el filtrado y la extracción, hasta obtener una función diferente g(x,y). La umbralización selecciona pixeles etiquetados como bordes b(x,y) y la localización suministra información exacta de la posición y orientación del borde y = p(x).

Al hablar de detección de bordes, el término sugiere que la aplicación de un algoritmo con este propósito dará como resultado un contorno. Sin embargo, el objetivo de un algoritmo de detección de bordes, es obtener imágenes cuya salida muestre pixeles de mayor intensidad en los valores que detecten transiciones cercanas. Los bordes son encontrados en zonas de la imagen donde el nivel de intensidad fluctúa bruscamente.

Cuanto más rápido se produce el cambio de intensidad, el borde es más fuerte. Para poder detectar los bordes de los objetos se deben detectar aquellos puntos de borde que los forman. Así, un punto de borde puede ser visto como un punto en una imagen donde se produce una discontinuidad en el gradiente. Un buen proceso de detección de bordes facilita la elaboración de fronteras de objetos, con lo que el proceso de reconocimiento de objetos se simplifica.

El método más aceptado en la detección de bordes, consiste en la aplicación de filtros de suavizado seguidos por filtros de derivadas. Estos últimos, son los que sirven para detectar los cambios o transiciones en los niveles de gris en una imagen.

A fin de lograr la localización de los puntos en los que se produce la variación de intensidad, se emplean métodos basados en los operadores derivada. Básicamente se

tienen dos posibilidades: aplicar la primera derivada (gradiente[3]) o la segunda derivada (laplaciana). En el primer caso se buscarán grandes picos, y en el segundo, pasos de respuesta positiva a negativa o viceversa.

A continuación se describen algunas de las propiedades de la primera y la segunda derivada en la detección de bordes:

La primera derivada produce un resalte de las zonas en que la intensidad no es homogénea.

Requisitos de la primera derivada:

- Debe ser cero en segmentos planos.
- Debe ser diferente de cero al inicio de escalones o rampas.
- Debe ser diferente de cero en las rampas.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Ecuación 1 – La primera derivada.

La segunda derivada origina un cambio de signo en la posición de borde. "Zero-crossing" (paso por cero).

Requisitos de la segunda derivada:

- Debe ser cero en áreas planas.

[3] Refiérase a ésta dirección:
http://140.148.3.250/u_dl_a/servlet/mx.udlap.ict.tales.html.Block?Thesis=200&Type=T

- Debe ser diferente de cero al inicio y al final de escalones o rampas.
- Debe ser cero en rampas de pendiente constante.

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Ecuación 2 – La segunda derivada.

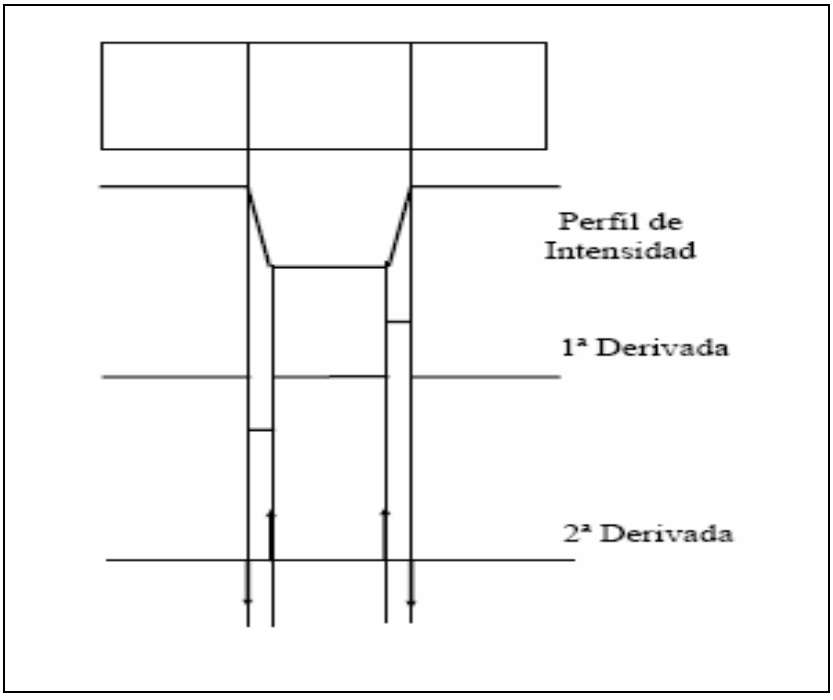


Figura 12 - Grafica del comportamiento de la 1º y 2º Derivada.

En la Figura 12 se puede observar gráficamente el comportamiento de la 1ª y 2ª derivada. En ella se pueden ver como es el cambio de la grafica según el perfil de intensidad; en la 1ª derivada se observa un cambio de alto a bajo si la pendiente del perfil va en decremento, por otro lado, si esta pendiente es creciente, la grafica de la 1ª derivada cambiara de bajo a alto. Con respecto a la 2ª derivada, sucede lo mismo, con la diferencia que este tendrá impulsos, estos dependerán si la pendiente es positiva o negativa.

1.2.1 El Gradiente.

El gradiente está formulado a partir del modelo de la primera derivada.

En procesamiento de imágenes, lo que se utiliza es la magnitud del gradiente.

El gradiente de *f*, en las coordenadas (x,y), se define como un vector columna de la siguiente manera:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Ecuación 3 – El Gradiente.

La magnitud de este vector está dada por:

$$\nabla f = mag(\nabla f)$$
$$= \left[G_x{}^2 + G_y{}^2\right]^{\frac{1}{2}}$$
$$= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}^2\right)\right]^{\frac{1}{2}}$$

Ecuación 4 – Magnitud del Gradiente.

Las componentes del vector gradiente son operadores lineales, pero su magnitud no lo es.

Una aproximación práctica de la magnitud del gradiente es la siguiente:

$$\nabla f \approx \left|G_x\right| + \left|G_y\right|$$

Ecuación 5 – Aproximación de la magnitud del gradiente.

Los detectores de bordes utilizan operadores ó máscaras de gradiente, cada detector de borde posee máscaras distintas que los diferencian.

Algunos de los algoritmos de detección de bordes más comunes son los siguientes:

- **Técnicas basadas en el gradiente:**
  - Operador de Roberts.
  - Operador de Sobel.

- Operador de Prewitt.
- Operador Isotrópico (u operador de Frei-Chen).

- **Operadores basados en cruce por cero:**

- Operador de Marr-Hildreth.
- Detector de Canny.

Los operadores basados en el gradiente asumen que los bordes de una imagen son pixeles con un alto gradiente. Un rápido índice de cambio de intensidad en alguna dirección dada por el ángulo del vector gradiente puede observarse en los pixeles de los bordes. En la Figura 13 se muestra un píxel de borde ideal con su correspondiente vector de gradiente. En el píxel, la intensidad cambia de 0 a 255 en dirección del gradiente. La magnitud del gradiente indica que tan marcado está el borde. Si se calcula el gradiente en regiones uniformes se obtiene un vector de valor 0, lo que significa que no hay pixeles de borde.

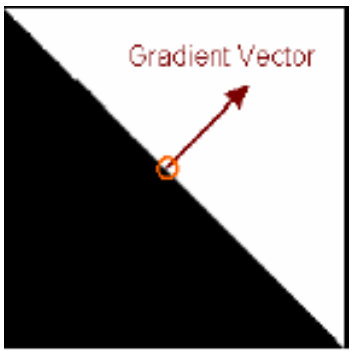


Figura 13 - Píxel de borde ideal.

Un píxel de borde se describe mediante dos características importantes:

1. Intensidad del borde, que es igual a la magnitud del gradiente.
2. Dirección del borde, que es igual al ángulo del gradiente.

Matlab tiene seis métodos diferentes de detección de contorno: Método Sobel, Método Prewitt, Método Roberts, Método Cruce por Cero, Método Laplaciano-Gaussiano y Método Canny. Los tres primeros utilizan una aproximación al operador derivativo (aproximación Sobel, aproximación Prewitt , y aproximación Roberts) y detectan contornos en los puntos donde el gradiente de la imagen es máximo. El método Cruce por Cero filtra la imagen y posteriormente busca cruces por cero, a los cuales interpreta como contornos. El método Laplaciano-Gaussiano[4] encuentra los bordes al buscar cruces por cero después de filtrar la imagen con un Laplaciano del filtro Gaussiano.

El gradiente se estima por medio del uso de operadores. Algunos de estos operadores se describen a continuación.

- Operador de Roberts.

Es el operador de gradiente más simple. Utiliza las direcciones diagonales para calcular el vector gradiente mediante las máscaras que se muestran en la siguiente figura:

[4] Refiérase a ésta dirección:
http://140.148.3.250/u_dl_a/servlet/mx.udlap.ict.tales.html.Block?Thesis=200&Type=T

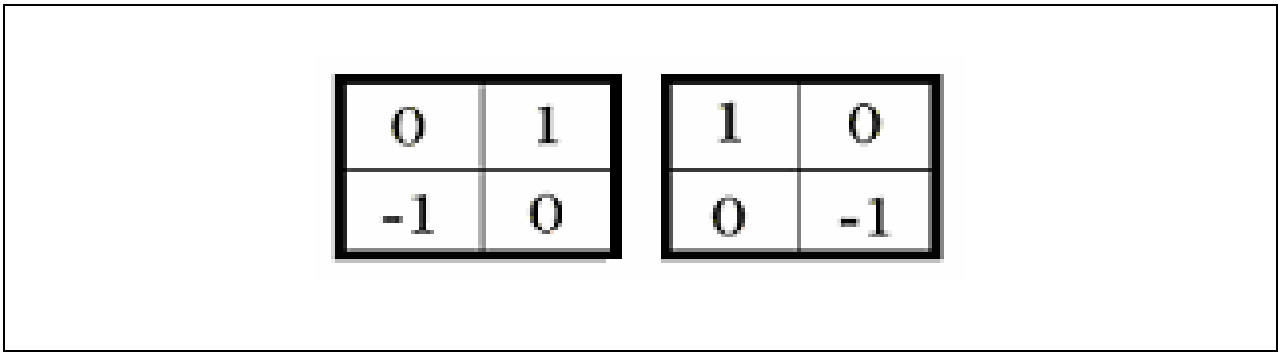


Figura 14 - Máscara Roberts

- Operador de Prewitt.

Acción compuesta de las máscaras:

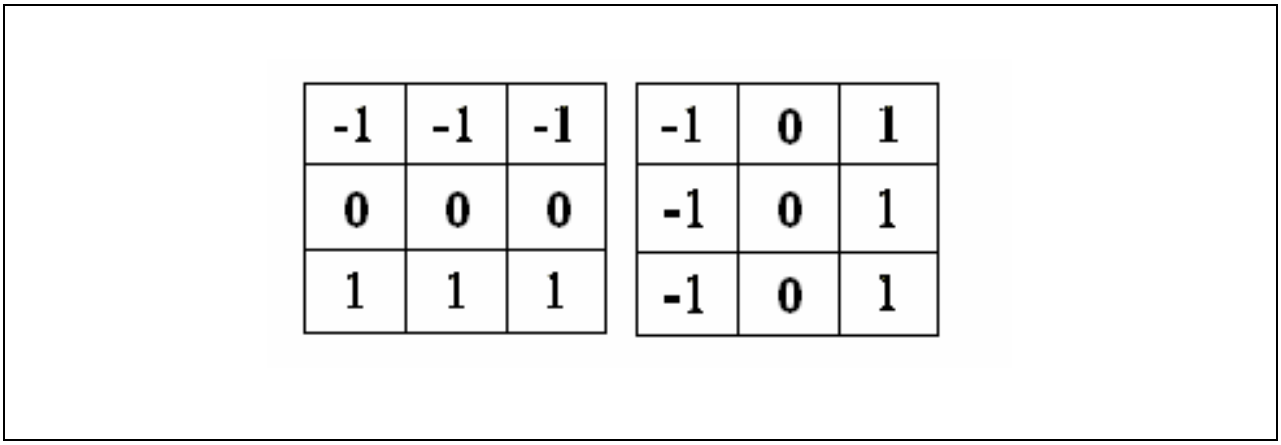


Figura 15 - Máscara Prewitt.

La imagen final se puede formar a partir de la convolución de la imagen inicial con cada una de las máscaras que se muestran en la Fgura 15 (resultados parciales), de alguna de las siguientes maneras:

- Raíz cuadrada del cuadrado de los resultados parciales.
- Suma de los valores absolutos de los resultados parciales.
- Máximo de los valores absolutos de los resultados parciales.
- Considerando sólo un resultado parcial.

La convolución de la imagen con cada máscara puede originar valores superiores a los de la imagen inicial, que pueden desvirtuar los resultados obtenidos. A tal fin, y siempre que no existan saturaciones en la imagen (por ejemplo, se permiten valores superiores a 255), se suele dividir el resultado final por la mitad de la suma de los valores absolutos de la máscara (en este caso es igual a 3). Si existen saturaciones en la imagen (por ejemplo, cualquier valor superior a 255 se convierte en 255) es conveniente dividir la imagen inicial por dicho número antes de convolucionarla con las máscaras.

- Operador de Sobel.

Acción conjunta de las máscaras:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figura 16 - Máscara Sobel.

Cada una de ellas se forma por la convolución de una componente del gradiente con el filtrado de la media ponderada del entorno.

La imagen final se puede formar a partir de la convolución de la imagen inicial con cada una de las máscaras de la Figura 16 (resultados parciales), de alguna de las siguientes maneras:

- Raíz cuadrada del cuadrado de los resultados parciales.
- Suma de los valores absolutos de los resultados parciales.
- Máximo de los valores absolutos de los resultados parciales.

- Considerando sólo un resultado parcial.

La convolución de la imagen con cada máscara puede originar valores superiores a los de la imagen inicial, que pueden desvirtuar los resultados obtenidos. A tal fin, y siempre que no existan saturaciones en la imagen (por ejemplo, se permiten valores superiores a 255), se suele dividir el resultado final por la mitad de la suma de los valores absolutos de la máscara (en este caso es igual a 4). Si existen saturaciones en la imagen (por ejemplo, cualquier valor superior a 255 se convierte en 255) es conveniente dividir la imagen inicial por dicho número antes de convolucionarla con las máscaras.

1.2.2 Método Canny.

El Método Canny tiene por definición menor probabilidad que los otros de ser engañado por el ruido y por ello resulta más eficaz para detectar bordes débiles verdaderos.

El método Canny es un proceso multi-etapa que se inicia con una convolución Gaussiana para eliminar el ruido y los detalles. La convolución es una operación matemática fundamental para el procesamiento de imágenes dado que ofrece una manera de realizar operaciones con matrices que usualmente son de diferentes tamaños, pero que se ubican en las mismas coordenadas espaciales, ya que describe el comportamiento de cualquier sistema lineal invariante y continuo en el tiempo. La convolución de dos funciones *h(t)* y *x(t),* denotado por *y(t)* = *h(t)*x(t)*, se define por la integral:

$$y(t) = \int_{-\infty}^{\infty} h(v)x(t-v)dv$$

Ecuación 6 – Convolución de dos funciones.

Donde *v* es una variable de integración.

Usualmente uno de los arreglos de entrada es una imagen de intensidad, el segundo arreglo puede ser uni o bi-dimensional, a éste arreglo se le denomina operador. En el caso de la convolución Gaussiana el operador representa la forma de una campana Gaussiana. En una dimensión la distribución Gaussiana tiene la forma :

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x^2}{2\sigma^2}}$$

Ecuación 7 – Distribución Gaussiana.

Donde σ es la variación estándar de la distribución. Se asume que la distribución tiene una media igual a cero.

Puesto que la imagen es digital y los valores de los píxeles son discretos, se necesita discretizar la función Gaussiana antes de llevar a cabo la convolución. En teoría, la distribución Gaussiana nunca es cero, lo cual requiere un operador infinitamente largo, pero en la práctica, el cero se encuentra a tres desviaciones estándar a partir de la media y así se pueden reducir las dimensiones del operador. La ventaja que presenta el operador surgido de la distribución Gaussiana es que actúa como un punto de dispersión de la convolución. La convolución se lleva a cabo rápidamente puesto que la distribución Gaussiana puede ser separada en sus componentes espaciales, así, la convolución bidimensional se lleva a cabo primero en una dimensión en la dirección de *x* y luego en la dirección de *y.*

Después de la convolución Gaussiana, la imagen pasa por un operador derivativo de dos dimensiones para resaltar las regiones de la imagen con una primera derivada espacial.

Los contornos de los objetos aumentan la pendiente del gradiente de magnitud de la imagen. Por ello el algoritmo sigue las pistas de éstos márgenes y fija todos los pixeles que no están en el punto máximo en cero para obtener a la salida una línea fina. Este proceso se conoce como supresión de los no-máximos.

1.2.3 Operador de Canny.

Se basa en el empleo del gradiente de la imagen filtrada con una función Gaussiana, en la Figura 17 se ilustra el proceso general de la obtención del gradiente.

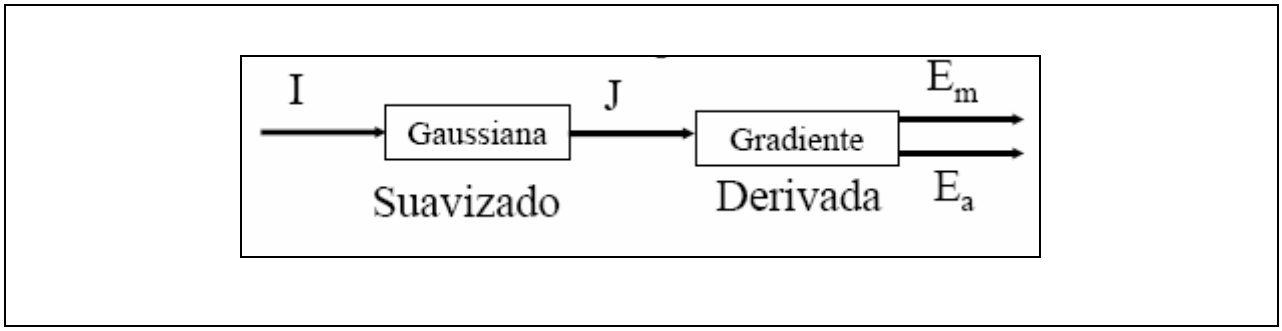


Figura 17 - Proceso general para la obtención del gradiente.

El algoritmo de canny efectúa el siguiente proceso:

- Supresión no máxima al resultado del gradiente
  - Para todo punto se obtiene la dirección más cercana dk a 0º, 45º, 90º y 135º en Ea(i,j).
  - Si Em(i,j) es menor que uno de sus dos vecinos en la dirección dk, $I_N(i,j)=0$. Si no $I_N(i,j) = E_m(i,j)$.

- Histéresis de umbral a la supresión no máxima.

  Permite eliminar máximos procedentes de ruido, etc.

  - Entrada I_N, Ea, y dos umbrales T_1 y T_2 (T_2>T_1)
  - Para todo punto en I_N, y explorando en un orden:
    - Localizar el siguiente punto tal que $I_N(i,j) > T_2$
    - Seguir las cadenas de máximos locales a partir de $I_N(i,j)$ en ambas direcciones perpendiculares a la normal al borde siempre que I_N>T_1. Marcar los puntos explorados.
    - La salida es un conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación.

- Cierre de contornos abiertos (Algoritmo de Deriche y Cocquerez)
  - La imagen de entrada es una imagen de contornos binarizada (1 = borde; 0 = no borde).
  - Para cada punto de borde de un extremo abierto se le asigna un código, ver Figura 18, que determina las direcciones de búsqueda para el cierre del contorno.

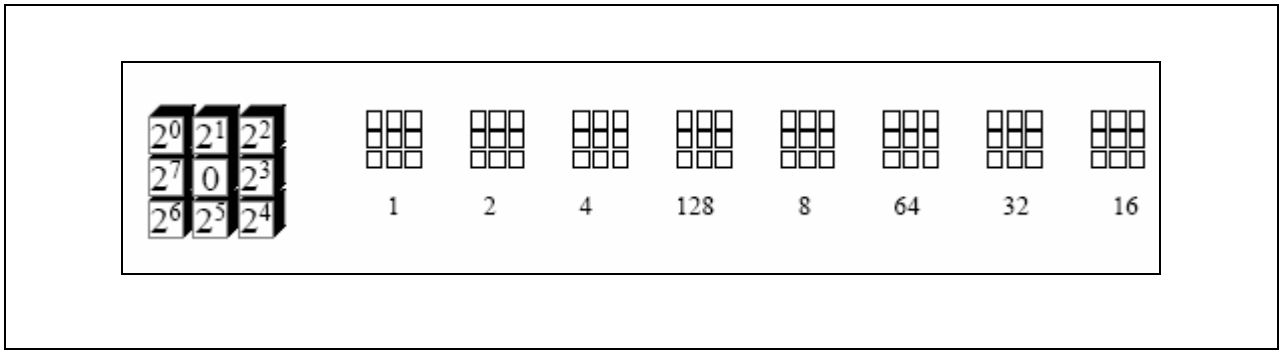


Figura 18 - Forma de asignación de códigos.

  - Para los pixeles marcados con este código se marca como pixel de borde el de máximo gradiente en las tres direcciones posibles.
  - Se repiten los pasos hasta que se cierren todos los contornos.

1.2.4 Operador Laplaciana.

Convolución con la máscara:

0	1	0
1	-4	1
0	-1	0

Figura 19 - Máscara Laplaciano, no se emplea filtrado del ruido.

También se emplea la convolución con la máscara:

1	1	1
1	-8	1
1	1	1

Figura 20 - Máscara formada por laplaciano en cuatro ejes (X-Y-R-C).

1.3 Procesamiento Morfológico.

La palabra morfología refiere al estudio de forma y estructura. En el procesamiento de imágenes, morfología es el nombre de una metodología específica originada en su estudio de materiales porosos. El nombre es adecuado ya que su análisis se basa en la estructura geométrica inherente en una imagen.

El análisis morfológico se realiza en una imagen de dos dimensiones en términos de alguna forma geométrica predeterminada conocida como elemento estructurante. Esencialmente, se estudia el modo en que el elemento estructurante está contenido en la imagen.

Por ejemplo, en la Figura 21 se ve una imagen S. Un cuadrado del tamaño ilustrado está contenida en la imagen si se posiciona su centro en el punto (1,2), pero no entra si se posiciona su centro en (3,1). Es claro que la forma en que el cuadrado entra en S según cambie su posición en el plano refleja la relación existente entre la estructura geométrica de S y la del cuadrado.

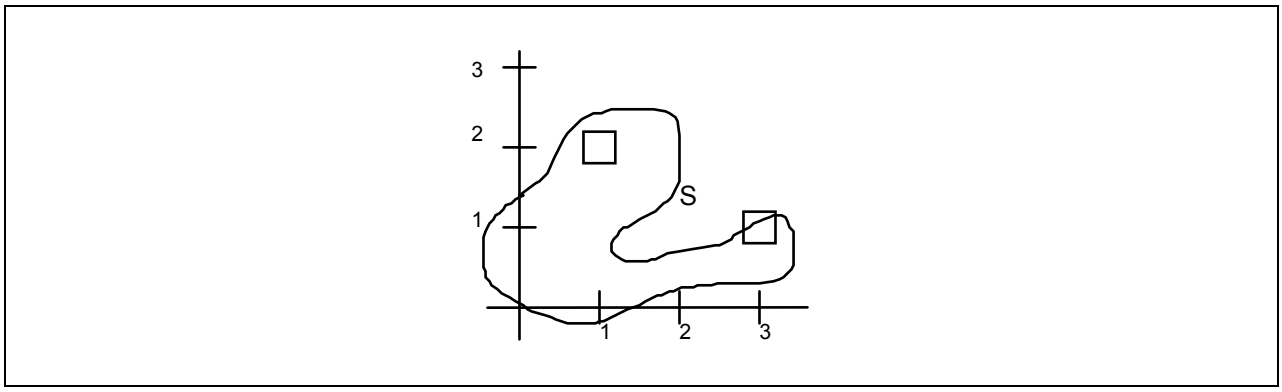


Figura 21 - Imagen S en un plano de 2 dimensiones.

Las operaciones morfológicas se pueden utilizar para diferentes fines, teles como, la detección de bordes, la segmentación, y el realce. En base a las operaciones morfológicas, se pueden construir filtros morfológicos que pueden utilizarse en lugar de los filtros lineales estándar. Es importante decir que los filtros morfológicos, dejan la imagen igual a la original.

1.3.1 Morfología de una Imagen Binaria.

Una imagen binaria es aquella que tienen dos niveles, generalmente blanco y negro. Esto hace que pueda ser representada mediante conjuntos. Es decir, el conjunto de todos los píxeles blancos de una imagen blanco y negro constituyen una descripción completa de la imagen.

Un elemento estructurante es un subconjunto de puntos en Z^2, cuya representación en el plano tiene cierta forma y tamaño. El elemento estructurante se concibe como un simple parámetro de forma para los filtros morfológicos.

La idea básica es probar la imagen con un elemento estructurante y cuantificar el modo en que está contenido dentro de la imagen. En una determinada ubicación dentro de la imagen pueden pasar dos cosas: que el elemento estructurante esté contenido o que no lo esté. Marcando las ubicaciones en que está contenido obtenemos información estructural de la imagen. Esta depende de la forma y del tamaño del elemento estructurante. La característica de estar contenido depende de la relación de subconjunto.

Las operaciones morfológicas básicas que se estudiarán en detalle son:

- DILATACION
- EROSION
- APERTURA
- CERRADURA

- **DILATACION.**

La dilatación es una transformación morfológica que combina dos conjuntos usando la suma vectorial de elementos de un conjunto.

$$A \oplus B = \{a+b : a \in A \text{ y } b \in B\}$$

Ecuación 8 – Dilatación

Es decir, el conjunto de todos los posibles vectores es la suma de pares de elementos, uno perteneciente a A y el otro a B.

Ejemplo:

A = {(0,1), (1,1), (2,1), (2,2), (3,0)}
B = {(0,0), (0,1)}
A ⊕ B = {(0,1), (1,1), (2,1), (2,2), (3,0), (0,2), (1,2), (2,2), (2,3), (3,1)}

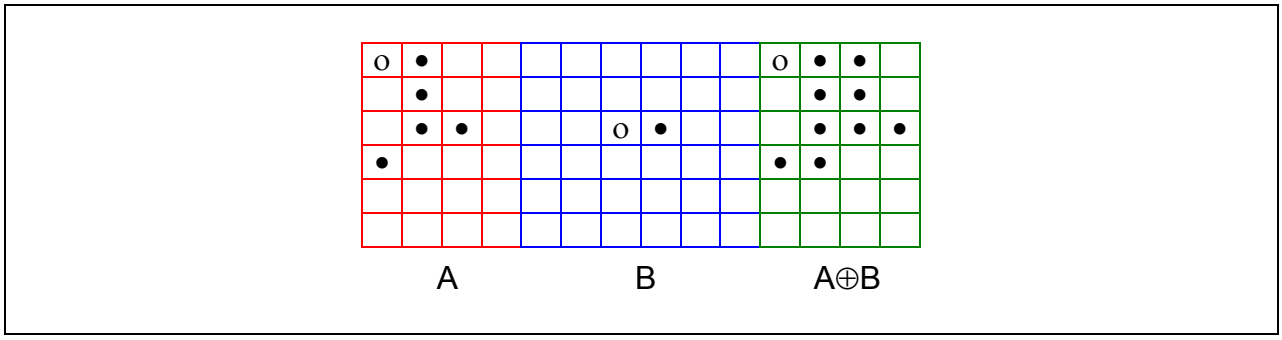
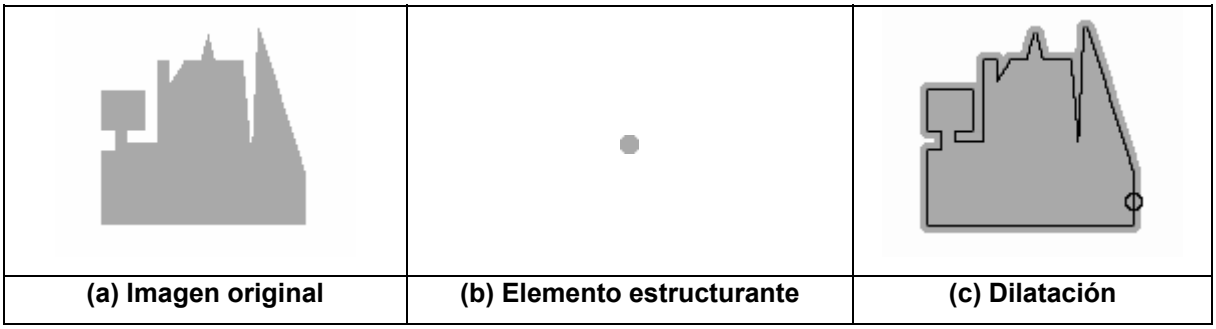


Figura 22 - Imagen que muestra el resultado de la suma de conjuntos, donde A es la imagen de entrada y B es el elemento estructurante, A⊕B es la suma de pares de elementos.

Ejemplo: Dilatación con un elemento estructurante circular, con el origen en el centro del mismo.



(a) Imagen original	(b) Elemento estructurante	(c) Dilatación

Figura 23 - Dilatación de una imagen.

- **TRASLACION.**

La traslación morfológica no es más que el desplazamiento de la imagen en el plano A_X, del conjunto A por el punto x.

$$\mathbf{Ax = \{a+x : a \in A\}.}$$

Ecuación 9 – Traslación.

Ejemplo:

A = {(0,1), (1,1), (2,1), (2,2), (3,0)}
X = (0,1)
$(A)_X$ = {(0,2), (1,2), (2,2), (2,3), (3,1)}

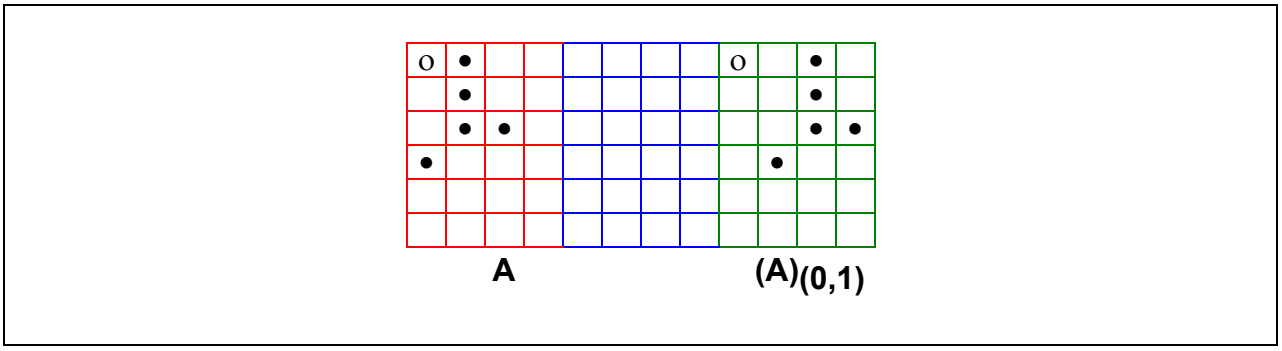


Figura 24 - Traslación de una imagen.

- **PRINCIPALES PROPIEDADES DE LA DILATACION**

  - **DILATACION COMO UNION DE TRASLACIONES.**

La dilatación de A por B puede ser calculada como la unión de traslaciones de A por los elementos de B.

$$A \oplus B = \cup_{b \in B} (A)b$$

Ecuación 10 – Dilatación como unión de traslaciones.

  - **CONMUTATIVA.**

La suma de la imagen de entrada más el elemento estructurante es igual a sumar el elemento estructurante más la imagen de entrada.

$$A \oplus B = B \oplus A$$

Ecuación 11 – Propiedad Conmutativa de la Dilatación.

- **ASOCIATIVA. (REGLA DE LA CADENA).**

Si una imagen A va a dilatarse por un elemento estructurante D, el cual a su vez puede ser expresado como la dilatación de B por C, entonces A⊕D puede calcularse como:

A ⊕ (B⊕C) = (A⊕B) ⊕ C ya que la suma es asociativa.

Ecuación 12 – Propiedad Asociativa de la Dilatación.

La forma **(A⊕B) ⊕ C** representa un considerable ahorro en la cantidad de operaciones que deben efectuarse cuando A es la imagen y B⊕C el elemento estructurante. El ahorro se produce porque la dilatación por fuerza bruta por B⊕C puede llevar operaciones del orden N^2 mientras que primero dilatando A por B y luego dilatar el resultado por C llevaría tanto como 2N operaciones, siendo N la cantidad de elementos en B y en C.

Ejemplo:

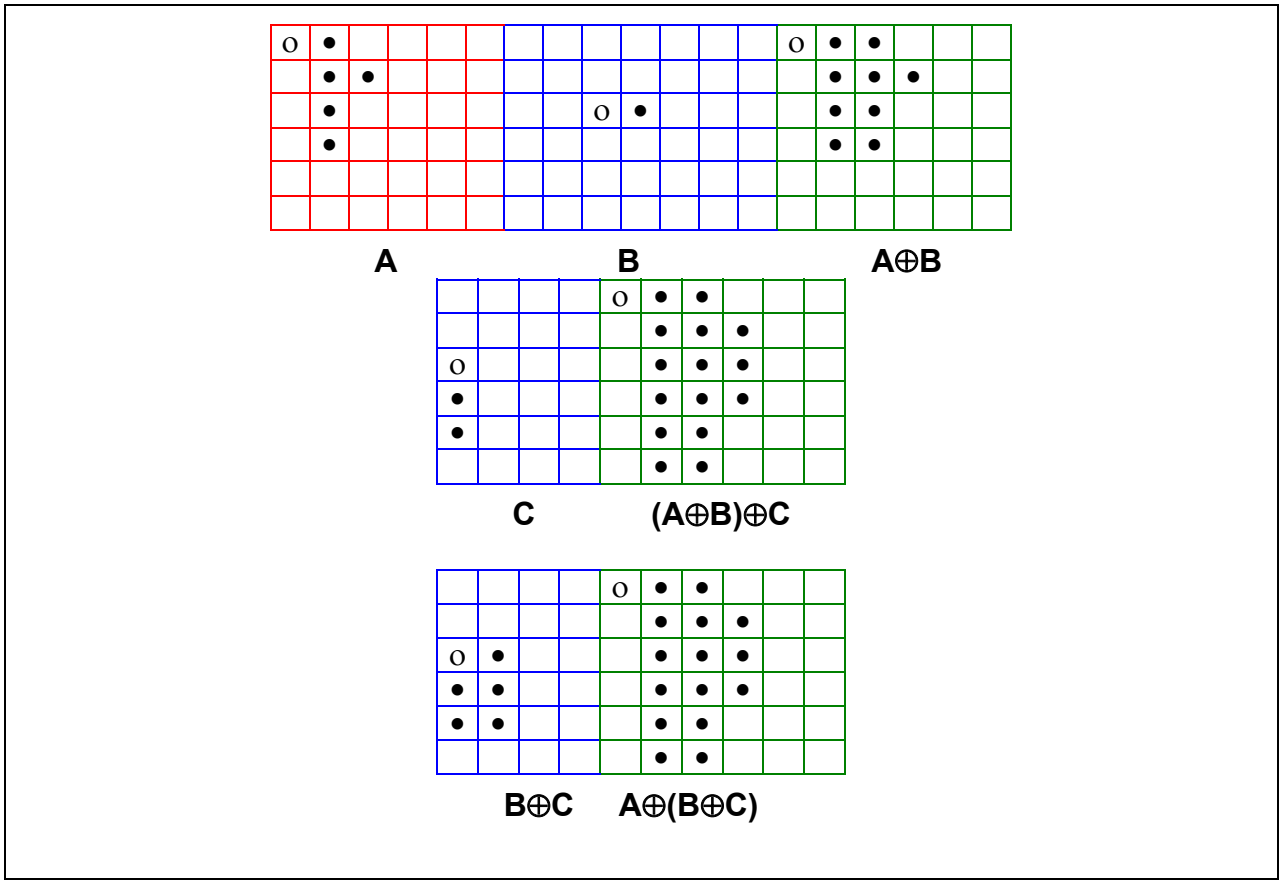


Figura 25 - Demostración gráfica de la propiedad asociativa.

- **INVARIANTE A TRASLACIONES.**

Si a una imagen A de entrada se traslada en el plano, y luego se suma un elemento estructurante, es lo mismo que sumar la imagen de entrada con el elemento estructurante y luego trasladar el resultado de esta suma.

Es decir:

$$(A)x \oplus B = (A \oplus B)x$$

Ecuación 13 – Invariante a traslaciones.

Ejemplo:

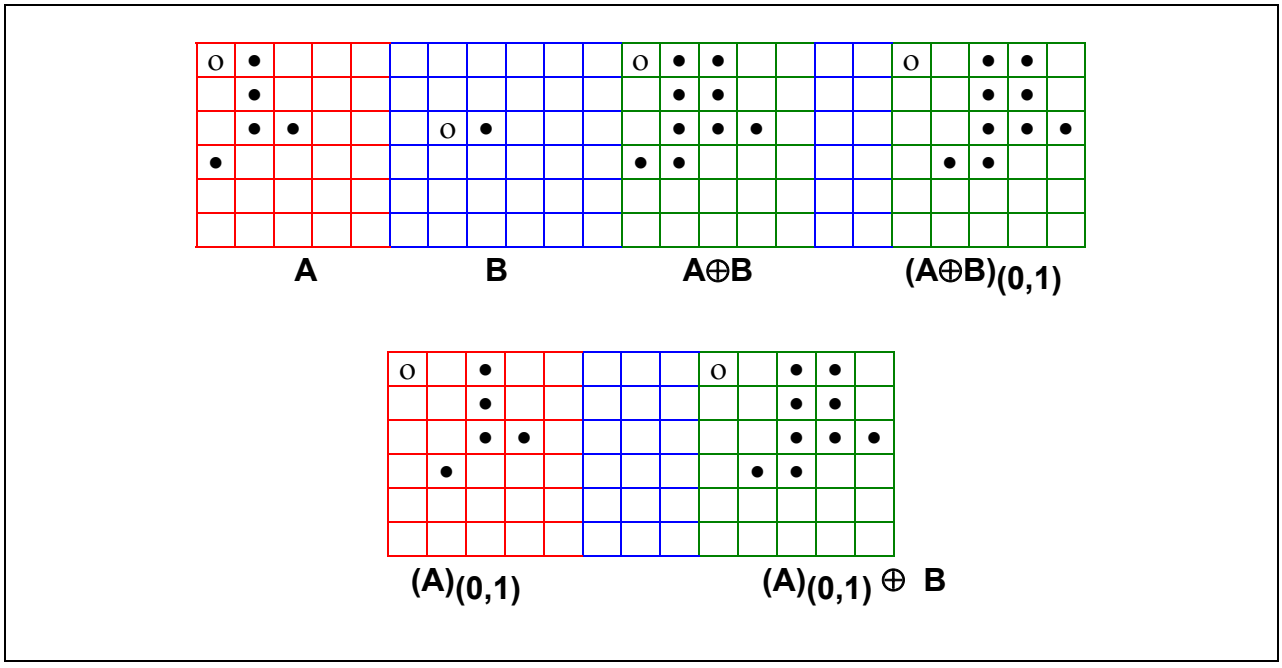


Figura 26 - Demostración gráfica de la propiedad de invariaciones a la traslación.

Esta propiedad es aplicada a dilataciones implementadas por la regla de la cadena. Aquí se establece que la traslación de uno de los elementos

estructurantes en la descomposición de la dilatación traslada la imagen dilatada en la misma forma.

$$A \oplus B_1 \oplus ... \oplus (B_n)_x \oplus ... \oplus B_N = (A \oplus B_1 \oplus ... \oplus B_n \oplus ... \oplus B_N)_x$$

Ecuación 14 – Regla de la cadena.

Una traslación en la imagen puede compensarse en la definición del elemento estructurante. En particular, sea el elemento estructurante B que compensa una traslación en la imagen A, tomando a B trasladado en la dirección opuesta. Entonces la traslación en B compensa la de A.

$$(A)_x \oplus (B)_{-x} = A \oplus B$$

Ecuación 15 – Compensación en la traslación.

Ejemplo:

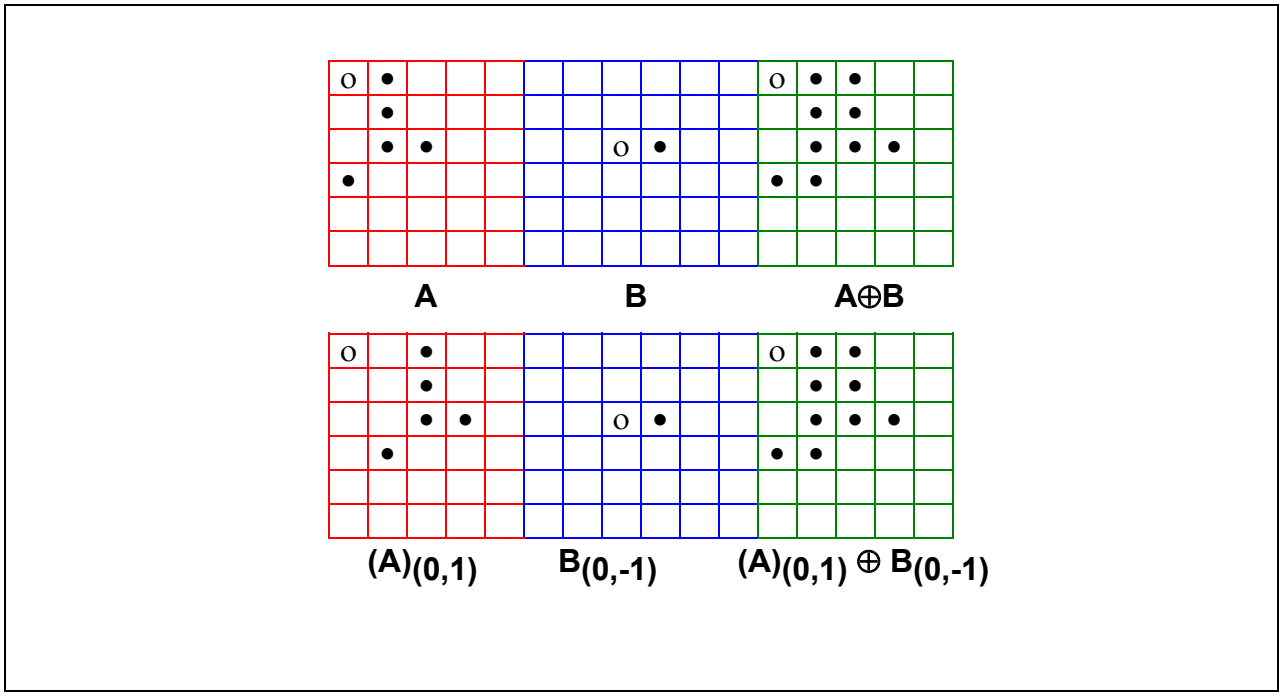


Figura 27.- Otro ejemplo de una imagen invariante a la traslación.

- **EXTENSIVA.**

Si el origen se encuentra dentro del elemento estructurante, la dilatación tiene el efecto de agrandar la imagen de entrada, obteniendo una imagen que incluye a la original. De no ser así, la imagen resultante puede no incluir a la original.

Si B contiene al origen, A ⊕ B0 no incluye A.

Ecuación 16 – Propiedad Extensiva

Ejemplo: Cuando el elemento estructurante B no contiene al origen, puede pasar que la dilatación de A por B no tenga nada en común con A.

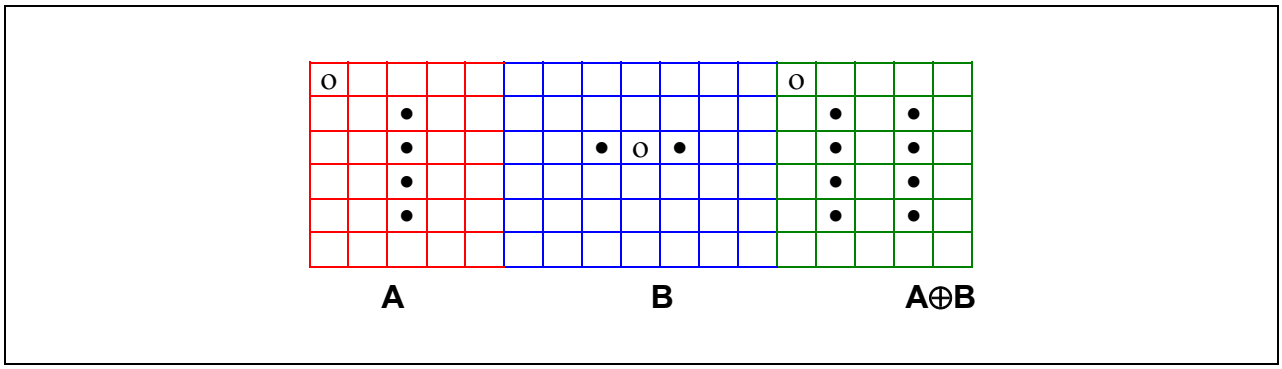


Figura 28 - Ejemplo gráfico de la propiedad extensiva.

- **LA EROSIÓN.**

Es la transformación que combina dos conjuntos usando la resta vectorial de elementos de conjuntos. Si A y B son conjuntos de dimensión N, entonces la erosión de A por B es el conjunto de todos los elementos x para los cuales x+b $\in$ A, para todo b $\in$ B.

$$A \ominus B = \{ x \in EN / x+b \in A \text{ para todo } b \in B \}$$

Ecuación 17 – Erosión.

o bien,

Es decir, son los puntos x para los cuales la traslación de B por x está contenida en A. La utilidad de esta transformación se aprecia mejor con la segunda definición.

Ejemplo:

A = {(1,0), (1,1), (1,2), (1,3), (1,4), (1,5), (2,1), (3,1), (4,1), (5,1)}
B = {(0,0), (0,1)}
AʙB = {(1,0), (1,1), (1,2), (1,3), (1,4)}

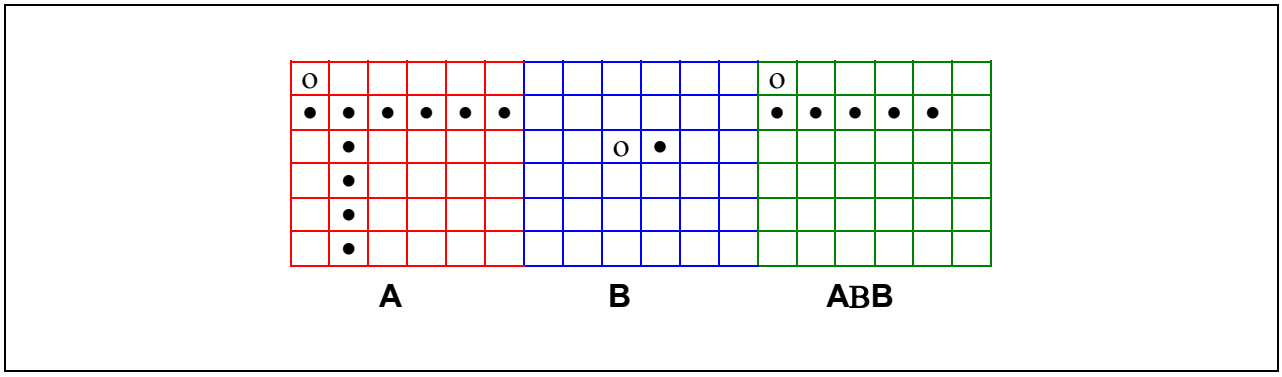
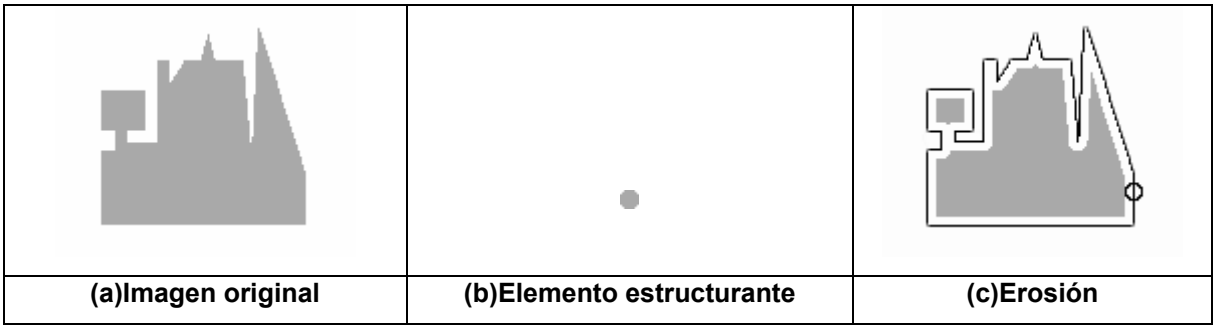


Figura 29 - Ejemplo gráfico de la erosión de una imagen.

Ejemplo:

Erosión con un elemento estructurante circular, con el origen en el centro del mismo.



(a)Imagen original	(b)Elemento estructurante	(c)Erosión

Figura 30 - Proceso de Erosión de una imagen

➢ APERTURA Y CERRADURA

En la práctica, la erosión y dilatación se utilizan en pares, la erosión seguida de dilatación o viceversa. En ambos casos, el resultado de aplicarlos iterativamente es la eliminación de detalle específico de la imagen menor al tamaño del elemento estructurante, sin una distorsión geométrica global de las características no suprimidas.

Por ejemplo, efectuar una apertura (erosión-dilatación) por un elemento estructurante con forma de disco, suaviza el contorno, rompe partes angostes, elimina pequeñas porciones y agudiza picos. Realizar una cerradura (dilatación-erosión) por una elemento estructurante con forma de disco, suaviza el contorno, une roturas angostas y golfos delgados, eliminando pequeños agujeros, y completa espacios en el contorno. La propiedad más saliente de estos filtros es que son independientes, es decir, la aplicación consecutiva de una apertura o cerradura no cambia el resultado de la primera aplicación. La importancia práctica es que constituyen estados completos y cerrados de algoritmos para el análisis de imágenes porque las formas pueden describirse en términos de los elementos estructurantes bajo los cuales pueden aplicarse aperturas o cerraduras sin efectuarse cambio alguno.

El origen del elemento estructurante no afecta el resultado de una apertura o de una cerradura.

El siguiente apartado ha sido extraído del libro:

REDES NEURONALES ARTIFICIALES Fundamentos, Modelos y Aplicaciones.
José R. Hilera/ Victor J. Martínez
Id. Biblioteca UDB: 001.535 H644 1995

De los cuales se han tomado los siguientes capítulos:
Capitulo I, II y III.

2 RECONOCIMIENTO DE PATRONES

En un principio el término de reconocimiento de patrones se refería originalmente a la detección de formas simples, tales como: escritos a mano, mapas del tiempo y espectros del lenguaje. El objetivo principal era el de imitar de una manera muy completa las funciones de los sistemas sensoriales biológicos.

Sus inicios, en 1960, estaban sostenidos sobre redes neuronales elementales, como Perceptron, Adaline y matrices de aprendizaje. En un principio les fue fácil su implementación, pero luego se observó que los sistemas biológicos eran muy difícil de alcanzar.

En el análisis de imágenes existen requerimientos que son muy difíciles de alcanzar, entre estos se tienen:

- Invariabilidad de la detección con respecto a la rotación, escala, perspectiva, oclusión parcial y captura simple de objeto, especialmente en condiciones de iluminación variables.

- Relación de observaciones de varios contextos a niveles diferentes de abstracción, en orden de distinguir los eventos más selectivamente.

Es importante destacar que hasta el sistema sensorial biológico más desarrollado no opera de forma autónoma, ya que la percepción sensorial está muy unida con el proceso cognitivo global. Cuando se da una replicación en las funciones sensoriales, no solo se imita el sistema sensorial, sino que se debe replicar todo el cerebro junto con sus capacidades de pensamientos, teniendo que precisar la capacidad de reconocimiento por el alto grado de aprendizaje.

Entre las áreas de aplicación más importantes del reconocimiento de patrones se tienen:

- Sensación remota.
- Análisis de imágenes médicas.
- Visión de computadoras industriales (especialmente para robots).
- Elementos de proceso de las entradas para computadoras.

Tareas concretas para las cuales se han desarrollado ya equipos de computadoras especiales:

- Segmentación y clasificación de imágenes.
- Reconocimiento de caracteres escritos (manuales y texto impreso).
- Reconocimiento del habla.
- Procesamiento y restauración de imágenes con ruido.

En un nivel más ambicioso, se puede lograr capacidades como:

- Análisis de imágenes (en lo referente a diferentes niveles temáticos de abstracción, tales como monitorización de uso terrestre en las bases de imágenes por satélite).
- Reconocimiento de imágenes (interpretación de escenas).
- Reconocimiento del habla (análisis e interpretación de frases habladas).

2.1 Redes Neuronales

Las redes neuronales son una tecnología muy utilizada para diferentes aplicaciones, éstas se pueden desarrollar en un período de tiempo razonable. Existen diferentes tipos de redes neuronales, las cuales tienen diferente aplicación, entre los campos que utilizan este tipo de tecnología se pueden mencionar: Biología, Empresa, Medio Ambiente, Finanzas, Manufactura, Medicina, Militares.

En su mayoría, las aplicaciones consisten en realizar un reconocimiento de patrones, es por ello que tienen que trabajar con datos sensoriales y de percepción, otros realizan filtrado o mapeo de las señales de entrada.

2.2 Tipos de Redes Neuronales.

En la siguiente tabla se muestra un resumen de los tipos de redes neuronales más conocidos en lo que respecta al reconocimiento de patrones, en ella se observan sus aplicaciones más importantes, sus ventajas e inconvenientes, sus creadores y el año en que fue desarrollada.

Nombre	**Año**	**Aplicación**	**Comentarios**	**Limitaciones**	**Creador(es)**
Perceptron	1957	Reconocimientos de caracteres impresos.	La red más antigua.	No puede reconocer caracteres complejos.	Frank Rosenbalt.
Backpropagation	1974-85	Reconocimiento de patrones, Síntesis de voz. Control de robots. Predicción.	Red más popular, Numerosas aplicaciones con éxitos facilidad de aprendizaje. Potente.	Se necesita mucho tiempo para el aprendizaje y muchos ejemplos.	Paul Werbos, David Parker, David Rumelhart.

Hopfield	1982	Reconstrucción de patrones y optimización.	Fácil de conceptuar.	Capacidad de estabilizar.	John Hopfield.
Teoría de resonancia adaptativa (ART).	1986	Reconocimiento de patrones (radar, sonar, etc.)	Sofisticada, poco utilizada	Sensible a la traslación, distorsión y escala	Gail Carpenter, Stephen Grossberg.

Tabla 1 - Redes neuronales utilizadas para el reconocimiento.

Las redes neuronales son aquellas que intentan reproducir el funcionamiento que desarrolla el cerebro humano, de la manera más exacta posible, por ello tienen mucho que ver con el estudio de la neurofisiología. Los principales investigadores que propusieron los modelos de neuronas son Warren McCulloch y Walter Pitts. El modelo está compuesto por entradas y salidas, las cuales son afectadas por pesos; donde la activación de cada neurona es igual a la suma de los productos de las entradas, siendo la salida una función de esta activación. La clave del sistema está en los pesos de las diferentes entradas, ya que las entradas son modificadas por los pesos y las salidas están en función de estas variaciones.

El entrenamiento de neuronas artificiales se realiza al tener muchas neuronas interconectadas, este entrenamiento consiste en aplicar diferentes entradas a la red, para después verificar la salida. En caso que la salida no tenga el resultado esperado, surge la variación de los pesos, los cuales son ajustados hasta llegar a obtener la salida deseada. Este entrenamiento debe tener varios elementos representativos para que la red vaya aprendiendo.

2.2.1 Neurona Biológica.

El cerebro humano está compuesto por una gran cantidad de neuronas, los estudios indican que por cada milímetro existen 50,000 neuronas; su tamaño y su

forma son variables pero manteniendo su estructura, la cual es mostrada en el siguiente esquema.

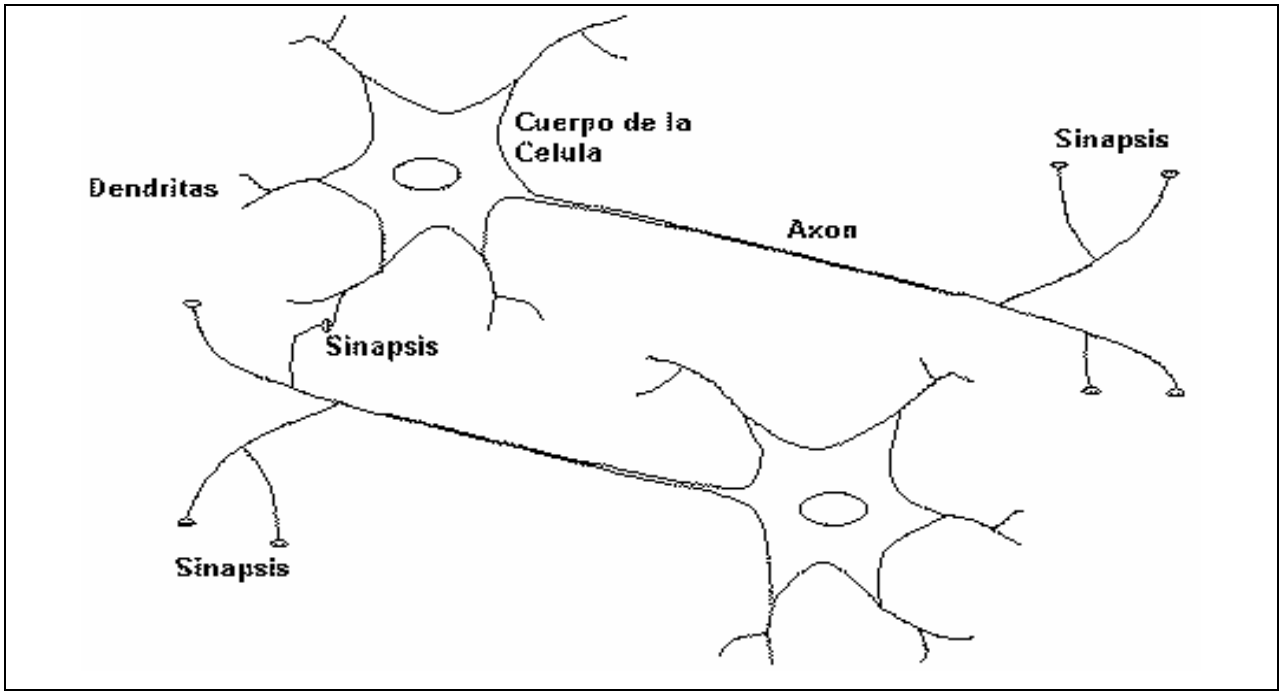


Figura 31 - Neurona biológica.

Los tres componentes principales de una neurona son:

- Las Dentritas.
- El cuerpo de la célula o soma.
- El Axón.

La Dentrita se encarga de recoger las señas enviadas por otras células a través de conexiones llamadas Sinápticas, si se relacionan las Dentritas con la neurona artificial, se puede decir que equivalen a la entrada de dicha neurona. El Axón es la salida de la neurona, la cual está interconectada con otras células nerviosas, enviando señales o impulsos. Cuando el Axón esta cerca de sus células destino, se divide en muchas ramificaciones las cuales se conectan con el Soma o Axónes de otras células, esta acción puede ser inhibidora o excitadora, según sea el

reconocimiento. Por otra parte, el Soma o cuerpo se encarga de todas las actividades metabólicas de la neurona y recibe la información de otras neuronas vecinas a través de las conexiones sinápticas.

2.2.2 Neurona Artificial.

Para comprender lo que es una neurona artificial, se relaciona un circuito electrónico, el cual desarrolle la suma de diferentes entradas y luego a la salida se obtenga un resultado que varía dependiendo del valor de las entradas, tal y como se muestra en la Figura:

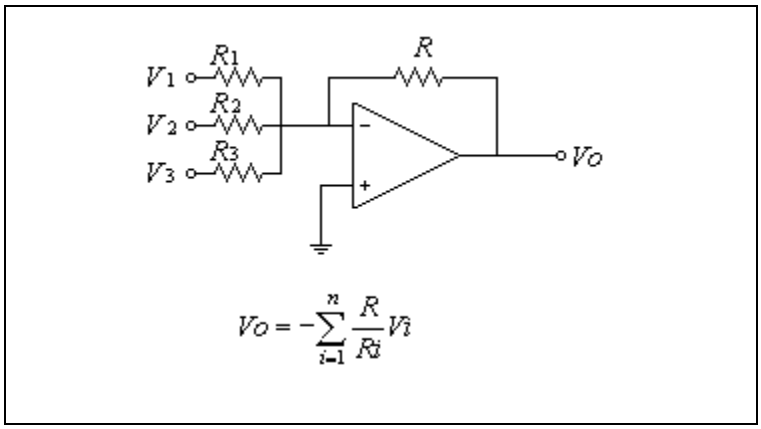


Figura 32 - .- Circuito representativo de una neurona.

Ahora se analiza la similitud de la neurona biológica con la neurona artificial.

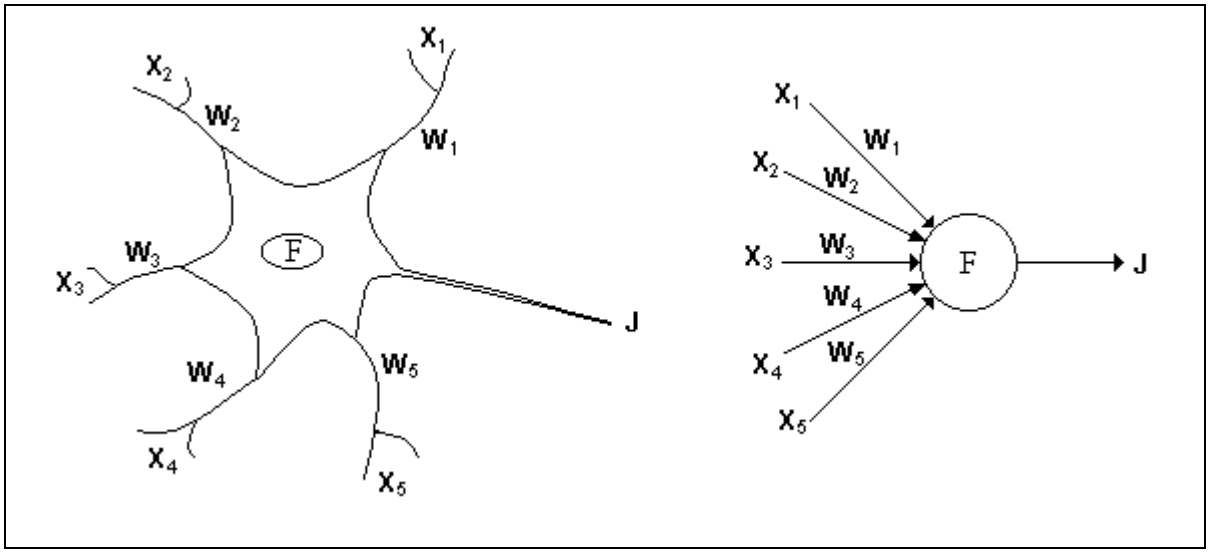


Figura 33 - Comparación de la neurona biológica y la neurona artificial.

Como podemos observar, la neurona artificial es la imitación de la neurona biológica, se comparan los diferentes parámetros de cada una de ellas:

- Las entradas X_i representan las señales que provienen de otras neuronas y que son capturadas por las Dendritas.

- Los pesos W_i son la intensidad de la sinápsis que conecta dos neuronas; tanto X_i como W_i son valores reales.

- F es la función umbral que la neurona debe sobrepasar para activarse; este proceso ocurre biológicamente en el cuerpo de la célula.

Las señales de entradas X_i en las neuronas artificiales, son variables continuas en vez de pulsos discretos, las señales de entradas pasan por ganancias o pesos W_i los cuales hacen la función de las funciones sinápticas que realizan las neuronas biológicas, estos pesos si son positivos equivalen a una acción excitadora y si son negativos equivalen a una acción inhibidora, por otro lado la función umbral significa la función de transferencia a la cual será sometida la neurona, matemáticamente la salida será igual a:

$$neta_i = \sum_{i=1}^{n} W_i X_1 = \vec{X}\,\vec{W}$$

Ecuación 18 – Función matemática de la salida de una neurona.

2.3 Conceptos Básicos de una Red Neuronal Artificial.

2.3.1 La neurona artificial.

El funcionamiento de una neurona artificial es simple, trata de recibir las entradas y calcular un valor a la salida. No importando el modelo a desarrollar es importante diferenciar tres tipos de unidades, estas se conocen como:

- Unidades de entradas.
- Unidades ocultas.
- Unidades de salidas.

Las unidades de entradas son las que reciben las señales provenientes de su alrededor, éstas pueden ser datos de sensores o de otro tipo de datos, éstas son la entrada de la red. Las unidades ocultas son aquellas las cuales poseen entradas y salidas, pero no tienen ningún contacto con el exterior, y las unidades de salida, son las que envían las señales fuera del sistema, es decir que son la salida de la red.

2.3.2 Estado de activación.

Es necesario saber los estados del sistema en un tiempo t. Esto se determina por un vector de números reales, los cuales representan el estado de activación del conjunto de unidades (entradas, ocultas, salidas), En el vector, cada elemento representa una unidad en un tiempo t.

$$A(t) = (a_1(t), a_2(t), a_3(t),\ldots\ldots a_N(t))$$

Ecuación 19 – Vector de estados de activación.

Las neuronas que componen la red están en cierto estado, dichos estados se conocen como: reposo y excitado, siendo estos dos los estados de activación. Estos pueden ser continuos o discretos, también pueden ser limitados o ilimitados.

Para comprender estos estados se hablará de un valor de activación discreto, estos pueden tomar un conjunto de valores binarios o pequeños, si se tiene un valor binario, el nivel lógico de 1 indicaría que el estado es activo, o sea excitado, y si el valor binario es un cero lógico indicaría un estado en pasivo, o sea reposo. Existen otros modelos los cuales tienen un conjunto continuo de estados de activación, en estos modelos se les asigna valores entre 0 y 1, o valores entre -1 y 1, dependiendo la función que se esté utilizando.

2.3.3 Función de transferencia.

En una red neuronal se encuentran un conjunto de conexiones que se unen entre sí, estas se conocen como unidades, cada unidad transmite señales a las que están conectadas a su salida, con cada unidad U se asocia una función de salida o de transferencia $F_1(a_1(t))$, la cual hace la transformación de un estado actual de activación $a_1(t)$, en una salida $y_1(t)$:

$$Y1(t) = F1(a1(t))$$

Ecuación 20 – Función de salida de un estado de activación.

Debido a la ecuación anterior, se puede decir que el vector que contiene la salida de la red en un instante t, queda de la siguiente forma:

$$Y(t) = (F1(a1(t)), F2(a2(t)),\ldots\ldots Fi(ait)),\ldots.FN(aN(t)))$$

Ecuación 21 – Vector de salida de red.

Existen diferentes tipos de funciones las cuales determinan los diferentes tipos de neuronas:

- Función escalón.
- Función lineal o mixta.
- Sigmoidal.
- Función Gaussiana.

- **Neurona de función escalón (hardlim).**

Esta función es utilizada cuando a las salidas solamente se necesitan dos posibles valores. Estas salidas se activarán solo cuando el estado de activación es mayor o igual a un valor de umbral de la neurona, esta función puede estar desplazada sobre los ejes. Este tipo de neurona es muy pocas veces utilizada, ya que sus capacidades están limitadas.

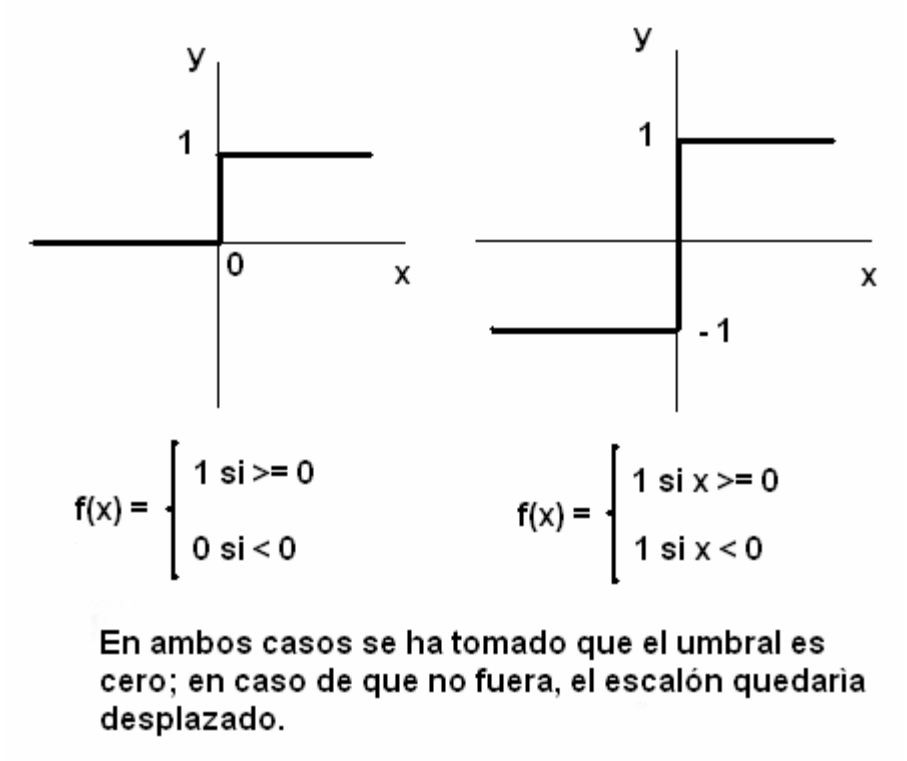


Figura 34 - Función Escalón.

- **Neurona de función lineal y mixta (purelin).**

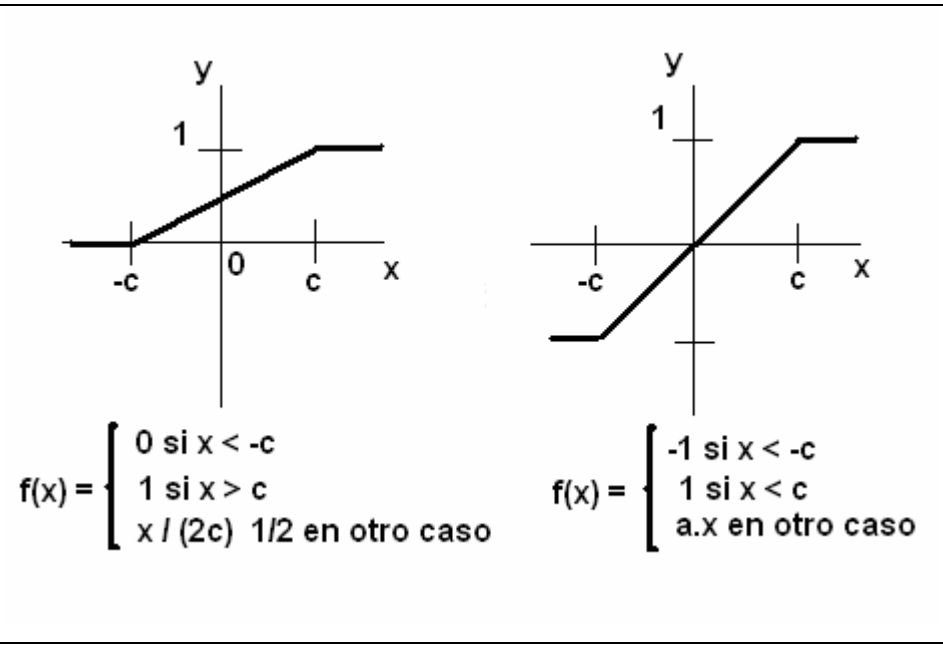


Figura 35 - Función lineal y mixta.

Esta función conocida como lineal o de identidad responde a la expresión matemática:

$$F(x) = x.$$

Ecuación 22 – Función Identidad.

En esta función, si la suma de las señales de entrada es menor que un límite inferior, la activación se define como 0. Caso contrario, si la suma es mayor o igual que el límite superior, la activación en este caso será 1. Por otro lado si la suma de las entradas está dentro del rango de estos límites, la activación está definida como una función lineal de la suma de todas las señales de entrada.

- **Neuronas de Función continua (sigmoidal).**

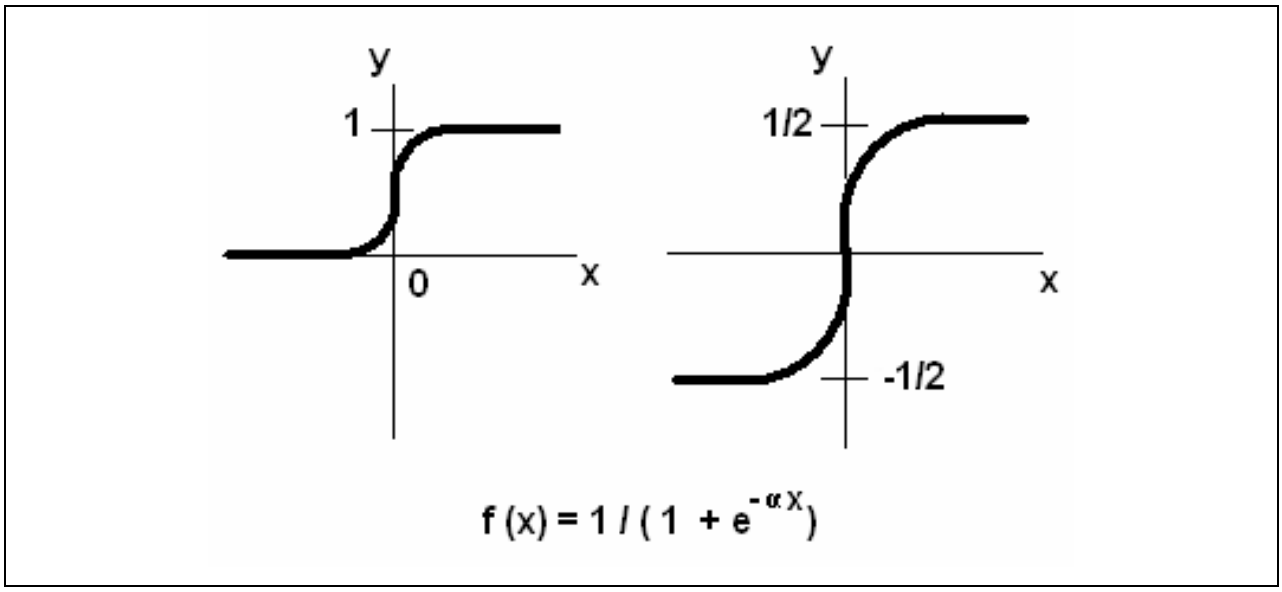


Figura 36 - Función contínua.

Toda función que esté definida en un intervalo de posibles valores de entrada, teniendo un incremento monótico y a la vez los límites superior e inferior, podrá realizar la función de transferencia de forma satisfactoria.

Con esta función, para la mayoría de valores del estímulo de entrada, el valor dado por la función tiene uno muy cercano a uno de los valores asintóticos, haciendo que en su mayoría su valor de salida esté comprendido en una de las dos zonas, alta o baja del sigmoide. Es por eso que si se tiene una pendiente elevada, ésta se comportará como la de la función escalón. Su importancia recae en que su derivada es siempre positiva o cercana a cero para valores grandes, sin importar su signo. Cuando x = 0 este toma su valor máximo.

Para esta función se pueden utilizar las reglas de aprendizaje de la función escalón, con la ventaja de que la derivada está definida en todo el intervalo. De no ser así, no ayudaría a los métodos de aprendizaje en los cuales se usan derivadas.

- **Función de transferencia gaussiana.**

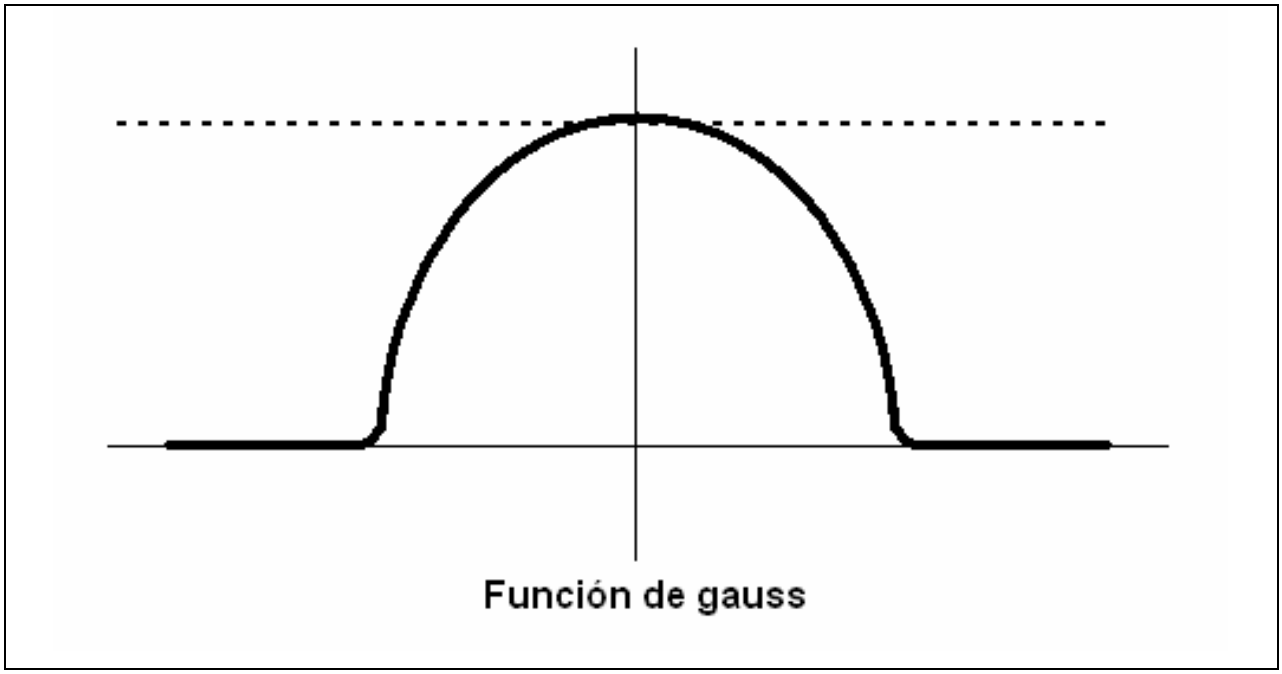


Figura 37 - Función Gaussiana.

En esta función, sus centros y su anchura pueden se adaptados, esto hace que sean más adaptativas que las funciones sigmoidales. Existen mapeos que suelen

requerir de dos niveles ocultos, utilizando neuronas con funciones sigmoidales; algunas veces se pueden realizar con un solo nivel en redes con neuronas de función gaussiana.

2.3.4 Conexiones entre neuronas.

Todas las conexiones que unen a las neuronas en una red neuronal artificial, tienen un peso, el cual hace que la red adquiera conocimiento. Si se considera y_i como un valor de salida de una neurona i en cierto tiempo dado, esta neurona recibirá un conjunto de señales que darán la información del estado de activación de todas las neuronas con la que se encuentra conectada. Cuando se conecta una neurona i con una neurona j, se pondera un peso W_{ij}. Es normal considerar que el efecto de cada señal sea aditivo, de tal forma que la entrada neta que recibe una neurona net_j es la suma del producto de cada señal individual por el valor de la sinapsis o conexión de ambas neuronas.

¡Error! No se pueden crear objetos modificando códigos de campo.

Ecuación 23 – Entrada neta de una neurona.

Esta regla muestra que el procedimiento a seguir para combinar los valores de entrada a una unidad con los pesos de las conexiones que llegan a esa unidad, se le conoce como regla de propagación.

Es común utilizar una matriz W con todos los pesos W_{ij}, estos indican la influencia de la neurona j sobre la neurona i. Esta matriz son números positivos, negativos o nulos. Si la matriz W_{ij} tiene un valor positivo, la interacción entre las neuronas i y j es excitadora, o sea que cuando la neurona j reciba una señal de la neurona i, esta última tendrá que activarla. Por otro lado si el valor de Wij es negativo, la interacción entre las neuronas i y j es inhibidora. En este caso, si la neurona i está activada, enviará una señal a j que tenderá a desactivar a ésta. Cuando el valor de la matriz W_{ij} es cero, esto indica que no hay conexión entre ambas.

2.3.5 Regla de activación.

Es necesario tener una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación. La función F, produce un estado nuevo de activación en una neurona, a partir del estado (a_i) que existía y la combinación de las entradas con los pesos de las conexiones (net_i).

Si se tiene el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega a ella, net i, el estado de activación que le sigue será $a_i(t+1)$, este se obtiene aplicando una función F, conocida como Función de activación.

ai(t+1) = F(a2t), Net2)

Ecuación 24 – Estado de activación en t+1.

Casi siempre F es la función identidad, es por eso que el estado de activación de una neurona en t+1 coincidirá con el Net de la misma en t. en este caso, el parámetro que se envía a la función F de la neurona, será directamente el Net. El estado de la activación anterior no se tiene en cuenta. O sea que la salida de una neurona i (y_i) quedará según la expresión:

¡Error! No se pueden crear objetos modificando códigos de campo.

Ecuación 25 – Salida de una neurona en un t+1.

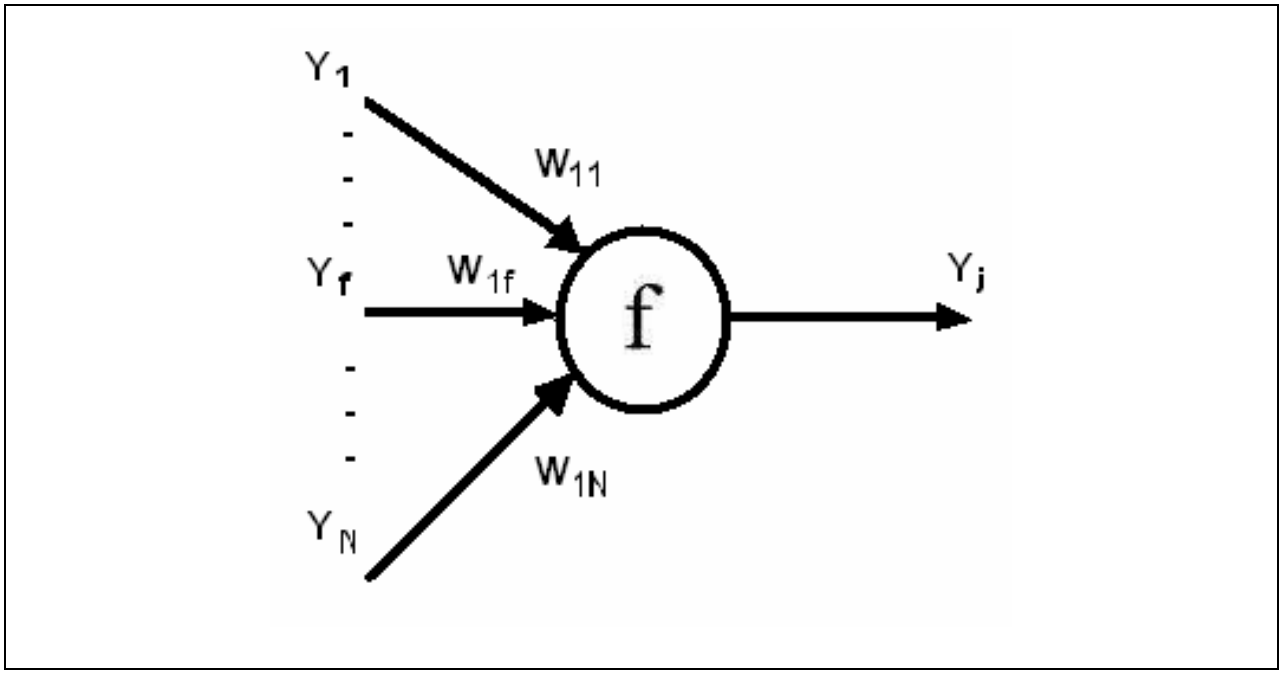


Figura 38 - Neurona artificial.

Seguidamente se considera únicamente la función F, que se denominará indistintamente como transferencia o de activación. Además la función de activación no está centrada en el origen del eje que representa el valor de la entrada neta, sino que está desplazada debido a las características internas de la propia neurona y que no es igual en todas ellas. Este valor se denota como θ_i y representa el umbral de activación de la neurona i.

¡Error! No se pueden crear objetos modificando códigos de campo.

Ecuación 26 – Salida de la neurona con umbral de activación.

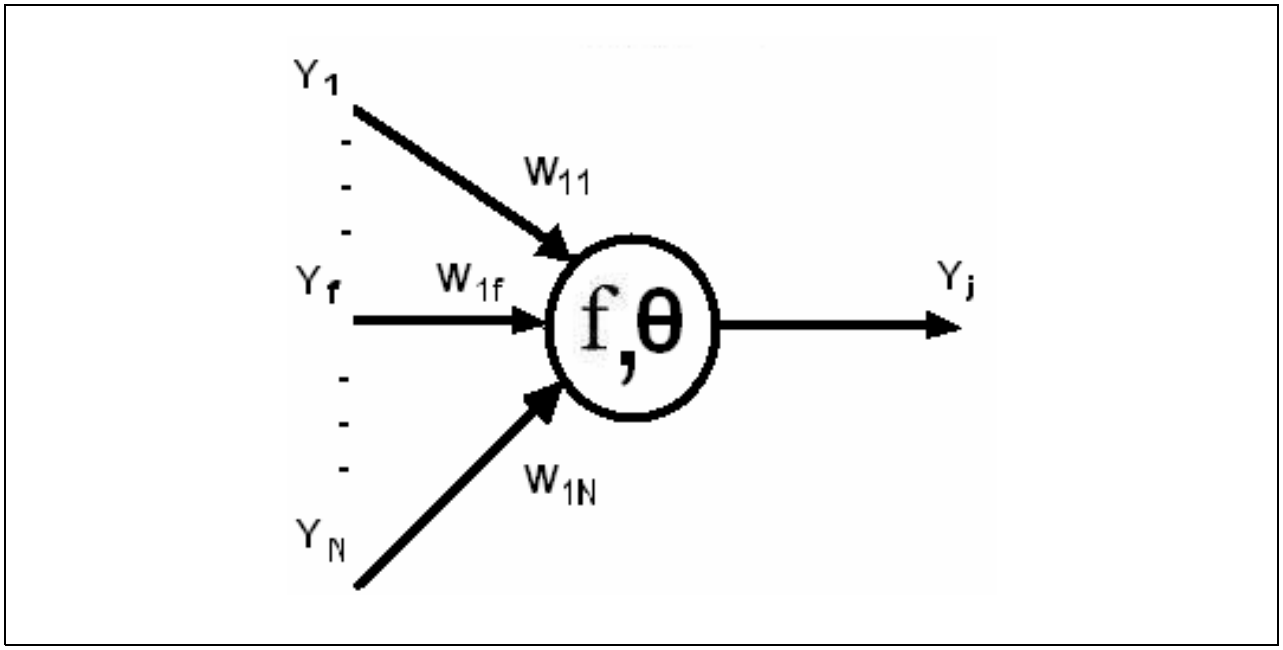


Figura 39 - Neurona artificial con umbral.

Las salidas obtenidas en una neurona para las diferentes formas de la función F, son:

- Función de activación escalón.
- Función de activación lineal o identidad.
- Función de activación lineal-mixta.
- Función de activación sigmoidal.

2.3.6 Regla de aprendizaje.

La regla de aprendizaje es la modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos.

En una neurona biológica el conocimiento se encuentra en las sinapsis, es decir, la información memorizada en el cerebro está mas relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas.

En la neurona artificial la memoria se encuentra en los pesos de las conexiones entre las neuronas. Una red artificial aprende al modificar los pesos de ella ya que este aprendizaje implica un cierto número de cambios en las conexiones.

Cada modelo o tipo red depende de sus propias técnicas de aprendizajes, su funcionamiento está dado por la cantidad de neuronas que ésta posea y el modo en que estén conectadas.

2.3.7 Representación Vectorial.

Es común utilizar modelos vectoriales para representar algunas magnitudes, si se toma en cuenta que la red está formada por varias capas de neuronas iguales, se puede decir que las salidas de cierta capa de n unidades se representan como vector n-dimensional Y = (y_1, y_2, y_3,….y_n). Si este vector n-dimensional de salida representa los valores de entrada de todas las unidades de una capa m-dimensional, cada capa poseerá una cantidad de n pesos asociados a las conexiones procedentes de la capa anterior, por lo que existen m vectores de pesos n-dimensionales asociados a la capa m.

El vector de los pesos de la j-ésima unidad tendrá la forma de:

$$\mathbf{Yj = (yj1, yj2, yj3…..yjn)}$$

Ecuación 27 – Vector n dimensional de pesos.

La entrada neta de la j-ésima unidad se puede escribir en forma de producto escalar del vector de entradas por el vector de peso. El producto está definido por

la suma de los productos de los componentes correspondientes de ambos vectores, hay que tomar en cuenta que los vectores tienen que tener la misma dimensión.

¡Error! No se pueden crear objetos modificando códigos de campo.

Ecuación 28 – Entrada neta de j-esima unidad.

n = representa el número de conexiones de la j-ésima unidad.

De acuerdo con lo anterior se utilizarán diferentes vectores, vectores de entrada, vectores de pesos, vectores de salida.

2.4 Estructura de una Red Neuronal Artificial.

Se ha venido hablando de los elementos de una neurona artificial, se mencionarán los más importantes:

- Unidades de procesamiento.
- Estado de activación de cada neurona.
- Patrón de conectividad entre neuronas.
- Regla de propagación.
- Función de transferencia.
- Regla de activación.
- Regla de aprendizaje.

Cada uno de los puntos anteriores están notados a los nodos que componen la red, a continuación se citará una red a una gran escala:

- Número de niveles o capas.
- Número de neuronas por nivel.

- Patrones de conexión.
- Flujo de información.

2.4.1 Niveles o capas de neuronas.

Una red neuronal artificial esta compuesta por un determinado número de neuronas las cuales pertenecen a tres diferentes capas, estas se conocen como:

- Capa de entrada.
- Capa oculta.
- Capa de salida.

Capa de entrada. Esta capa es la que recibe la información del exterior de la red.

Capa oculta. Es la que está entre la capa de salida y la capa de entrada, el número de capas puede ser de cero en adelante según sea necesario, estas capas no tienen contacto con el exterior, las capas ocultas pueden definir el tipo de red que se esté utilizando.

Capa de salida. Es la que tiene contacto hacia el exterior.

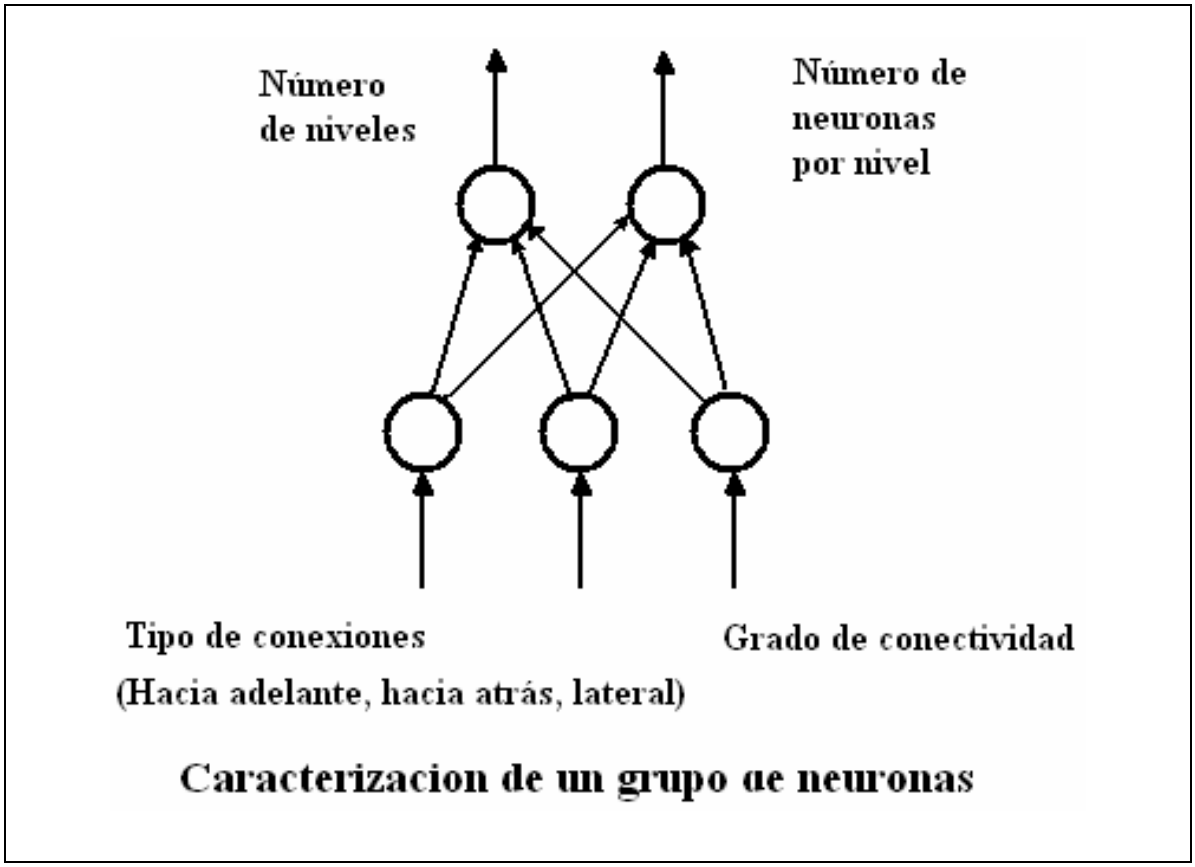


Figura 40 - Ejemplo de una red neuronal.

2.4.2 Formas de conexión entre neuronas.

Cuando se habla de conexiones entre nodos de una red neuronal es de mencionar la conectividad, esto se refiere a la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. Esta señal de salida de un nodo puede ser una entrada de otro elemento, incluso puede ser entrada de sí mismo, a este último se le conoce como conexión autorrecurrente.

Existen dos tipos de propagación en cuanto su forma de conexión:

- Propagación hacia delante.
- Propagación hacia atrás.

La propagación hacia delante se da cuando ninguna de las salidas de las neuronas es entrada del mismo nivel o de niveles anteriores a ésta, caso contrario sucede con las salidas de las neuronas que si pueden ser conectadas con neuronas del mismo nivel o de niveles anteriores a ésta, a éstos se les conoce como sistemas recurrentes.

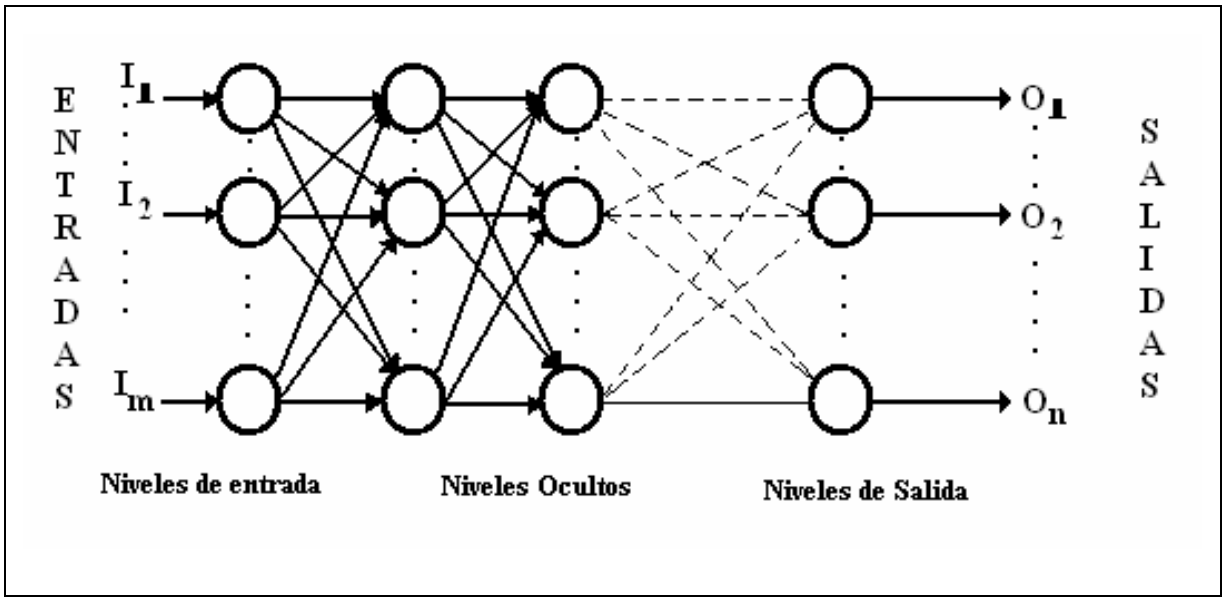


Figura 41 - Red neuronal con propagación hacia adelante.

2.5 Topologías de las Redes Neuronales.

Cuando se habla de topología de una red neuronal, se refiere a su forma de organización y disposición de las neuronas en la red, las cuales como ya se sabe están formadas por capas o agrupaciones de neuronas que están un tanto alejadas de la entrada y salida de la red. Es por ello que se pueden mencionar los siguientes parámetros que son fundamentales en una red:

- Número de capas.
- Número de neuronas por capas.

- Grado de conectividad.
- Tipo de conexiones de neuronas.

Al realizar la clasificación de las redes en sentido topológico éstos se pueden distinguir según su número de capas; éstas pueden ser de una sola capa o nivel de neuronas, o de redes con múltiples número de capas.

2.5.1 Redes Monocapa.

En las redes monocapa se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. Pueden existir redes monocapa autorrecurrentes, más sin embargo, algunas veces tal recurrencia no se utiliza, como el caso de la red hopfield.

La topología crossbar, es una topología equivalente a la de las redes de una capa, la red de tipo crossbar consiste en una matriz de terminales o barras que se cruzan en unos puntos a los que se les asocia un peso. Este tipo de topología es utilizada como etapa de transición cuando se pretende implementar físicamente una red monocapa, debido a que es relativamente sencillo desarrollar como hardware una estructura como la indicada.

Las redes monocapa son utilizadas comúnmente en tareas relacionadas con lo que se conoce como auto asociación, un ejemplo sería el de regenerar informaciones de entrada que se presentan a la red incompleta o distorsionada.

2.5.2 Redes Multicapa.

Este tipo de redes disponen de un conjunto de neuronas agrupadas en varios niveles o capas. Para distinguir a que capa pertenece una neurona, hay que fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida.

Es común que todas las neuronas de una capa reciban señales de entrada de otra capa anterior, más cercana a la entrada de la red, éstas envían las señales de salida a una capa superior, más cercana a la salida de la red. El nombre que recibe este tipo de conexiones es, conexiones hacia adelante o feedforward.

Por otro lado, existen redes en las cuales se pueden conectar las salidas de neuronas de capas posteriores a entradas de capas anteriores, a este tipo de conexiones se les conoce con el nombre de conexiones hacia atrás o feedback.

Como podemos observar las redes multicapa las podemos distinguir por sus dos tipos de conexiones, conexiones hacia delante o feedforward, y las de conexiones hacia atrás o feedback.

2.5.3 Redes con conexiones hacia delante.

En las redes con conexiones hacia delante o feedforward, las señales de las neuronas son propagadas hacia delante a través de las capas de la red. En este tipo de red no existen conexiones hacia atrás, conexiones recurrentes, ni laterales.

La red Backpropagation y la red Perceptron son dos de las redes que utilizan conexión hacia delante.

2.5.4 Redes con conexiones hacia atrás.

Las redes con conexión hacia atrás o feedback, tienen la peculiaridad de propagarse tanto hacia delante como hacia atrás, durante el funcionamiento de la red. Existen redes que suelen ser bicapa, por lo cual se encontrarán dos conjuntos de pesos: los que corresponden a la conexión feedforward de la primera capa hacia la segunda y los de las conexiones feedback de la segunda a la primera. En la mayor parte de los casos los valores de los pesos son diferentes.

El tipo de estructura bicapa es adecuado para realizar una asociación de información o patrón de entrada en la primera capa, con otra información o patrón de salida en la segunda capa, éste se le conoce como heteroasociación, aunque se pueden utilizar para la clasificación de patrones.

En algunos tipos de redes su funcionamiento está basado en lo que se conoce como resonancia, de manera que las informaciones de la primera y segunda capa interactúan entre sí hasta alcanzar el estado estable, permitiendo un mejor acceso a las informaciones almacenadas en la red. La red Art, es una de las redes que utiliza este tipo de conexión.

En el grupo de conexiones hacia adelante existen algunas que tienen conexiones laterales entre neuronas de la misma capa. Estas conexiones están diseñadas como conexiones excitadoras, permitiendo la cooperación de neuronas, o como inhibidoras, estableciendo una competición entre neuronas correspondientes. A este tipo de red que también dispone de conexiones autorrecurrentes se les denomina CABAM (Competitive Adaptive Bidireccional Associative Memory).

2.6 Mecanismo de Aprendizaje.

Cuando se habla de aprendizaje en una red neuronal, hablamos de modificar sus pesos debido a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. También los sistemas biológicos realizan una creación y destrucción de conexiones. En el caso de las redes artificiales, cuando se crea una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero. Por otro lado, una conexión se destruye cuando su peso pasa a ser cero.

Un proceso de aprendizaje termina cuando los valores de los pesos permanecen constantes, o sea, $dw_{ij}/dt = 0$, en este proceso los pesos de las conexiones sufren modificaciones hasta llegar a lograr la estabilidad. Algo bien importante en es conocer como los pesos van cambiando, es decir, que criterios se tienen que tomar en cuenta para cambiar el valor asignado a las conexiones cuando se desea que la red aprenda una nueva información. Estos criterios determinan lo que se conoce como regla de aprendizaje de la red. Se consideran 2 reglas principales: las primeras son las que responden a lo que se llama aprendizaje supervisado, y las segundas responden a lo que se conoce como aprendizaje no supervisado.

Según su aprendizaje, existen dos tipos de redes:

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

La principal diferencia entre estos dos tipos de redes está en la existencia o no de un agente externo que controle el proceso de aprendizaje de la red. También se puede utilizar para diferenciar las reglas de aprendizaje, la consideración de que si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red; es decir, inhabilitar el proceso hasta que el aprendizaje termine.

Según sea el caso, si es el primer caso se llamará ON LINE y si es el segundo caso sería OFFLINE.

El aprendizaje OFFLINE se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba, que serán utilizados en la correspondiente fase. En este tipo de aprendizaje los pesos de las conexiones permanecen constantes después que termina la etapa de entrenamiento de la red.

Estos sistemas no presentan inestabilidad en su funcionamiento debido a su carácter estático.

Por otro lado, el aprendizaje ONLINE no se distingue entre fase de entrenamiento y de operación, de manera que los pesos estarán variando según se presente nueva información al sistema. Es importante realizar un estudio de estabilidad debido al carácter dinámico de las mismas.

2.6.1 Redes con aprendizaje supervisado.

La característica más importante del aprendizaje supervisado, es que su entrenamiento está controlado por un agente externo el cual determina la respuesta que debería generar la red a partir de una entrada determinada. Dicho supervisor comprueba que la salida respecto de la entrada sea la deseada, de lo contrario procederá a modificar los pesos de las conexiones, con el propósito de llegar a obtener la salida deseada.

Existen tres tipos de aprendizaje supervisado:

- **Aprendizaje por corrección de error.**

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

Existe una versión del algoritmo backpropagation que se suele utilizar en redes multicapa que representan conexiones recurrentes con el fin de que estas redes aprendan la naturaleza temporal de algunos datos. Este tipo de redes de corrección de error también son utilizadas por algunas redes monocapa con conexiones laterales y autorrecurrentes.

- **Aprendizaje por refuerzo.**

Este tipo de aprendizaje es más lento que el aprendizaje por corrección de error, está basado en la idea de no disponer de un ejemplo completo del comportamiento deseado, o sea, no definir exactamente la salida que se desea ante una entrada determinada.

En este aprendizaje se da una indicación dependiendo de el valor de salida, se toma como éxito = +1 si la red fue ajustada, y se toma como fracaso = -1, si la red no fue ajustada. Si fuese este último caso, la red reajusta los pesos basándose en un mecanismo de probabilidades.

- **Aprendizaje estocástico.**

Este aprendizaje realiza cambios aleatorios de los pesos de las conexiones de la red y evalúa su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

En el aprendizaje estocástico se asocia la red neuronal con un sólido físico que tiene cierto estado energético. Este estado de energía se relaciona con el funcionamiento deseado de la red, es decir, si el funcionamiento es el adecuado se asume poca cantidad de energía, por lo que no será necesario ajustar los pesos. Si su funcionamiento no es el adecuado, se dice que la cantidad de energía es grande, por lo que será necesario ajustar los pesos.

Los cambios de energía de la red pueden ir disminuyendo si el comportamiento de la red se va acercando a lo esperado, por lo que se aceptará el cambio de los pesos; caso contrario, esta energía aumentará y el cambio de los pesos no será aceptado.

2.6.2 Redes con aprendizaje no supervisado.

Las redes con aprendizaje no supervisado son conocidas también como redes autosupervisadas, estas no poseen de una influencia externa para la modificación de sus pesos de las conexiones entre sus neuronas. La red no recibe ninguna información de si la salida del sistema es la esperada, por eso se dice que estas redes son capaces de auto organizarse.

Debido a que estas redes no poseen ningún supervisor, deben encontrar características, regularidades o categorías en las que se puedan establecer los datos que se presenten en la entrada del sistema. Se pueden asumir posibilidades en cuanto a la interpretación de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje empleado.

Muchas veces existe un grado de similitud en la salida, debido a la familiaridad entre los datos que se le presentan a la entrada y los datos que se han estado presentando, en otros casos se puede dar una clusterización o establecimiento de categorías, esto consiste en indicar a la salida de la red la categoría a la que

pertenece la información presentada a la entrada, cabe mencionar que es la propia red la que encuentra la categoría apropiada a partir de correlaciones entre las informaciones presentadas. Una variación de esta categorización es el prototipazo, es decir, la red obtiene prototipos que representan las clases a las que pertenecen las informaciones entrantes.

El aprendizaje sin supervisión realiza una codificación de los datos de entrada, esto hace que se genere una versión codificada de la entrada, con menos bits, pero manteniendo los datos importantes de la información.

Otra de las características de este tipo de redes es que realizan un mapeo de características, esta consiste en obtener un mapeo topográfico de los datos de entrada, En las redes con aprendizaje no supervisado existen dos tipos de aprendizaje, estos son:

- Aprendizaje hebbiano.
- Aprendizaje competitivo y cooperativo.

3 DISEÑO DE LA APLICACIÓN

3.1 Pruebas de Procesamiento de imagen.

3.1.1 Captura de imagen.

Las imágenes capturadas serán parte del brazo y la mano, realizando alguna de las señas del alfabeto internacional de lenguaje de sordos.

Al momento de capturar las imágenes, deben cumplirse ciertos requisitos del entorno que garanticen una calidad aceptable, asegurando la captura de las características más relevantes, con suficiente iluminación.

Debe procurarse:

- Fondo blanco. Debido a que la información importante es la mano, no tiene que haber ningún objeto detrás, en la Figura 42 (a) se muestra una imagen sin fondo, ésta introduciría ruido al procesamiento. Para evitar datos que no se desean, es necesario que la imagen posea un fondo Figura 42 (b), el cual ayude a obtener la información necesaria.

	
(a) Imagen sin fondo.	**(b) Imagen con fondo blanco.**

Figura 42.- Fondo de una imagen.

Como puede notarse, la eliminación de los objetos que rodean la mano, ayuda enormemente a segmentar la imagen, es decir separar el área de interés de objetos irrelevantes.

- Iluminación. Para resaltar la información necesaria que se requiere, es importante tener una buena iluminación, de modo que sean claramente identificables los rasgos o pliegues en la mano. En la Figura 43(a) se observa la imagen con escasa iluminación, esto no ayudaría a resaltar los datos importantes, ya que se generan sombras a los costados y en la mano. Por otro lado, el caso opuesto será tener mucha iluminación, ver Figura 43(b), esto producirá perdida de información, ya que la intensidad de luz genera espacios en blanco. La iluminación promedio, es decir ni mucha ni poca luz, es con la que mejores resultados se obtienen, ya que no se pierde información, ni se introducen datos no importantes para el procesamiento. La imagen de iluminación promedio se puede ver en la Figura 43(c).

		
(a) Poca iluminación.	**(b) Iluminación media.**	**(c) Mucha iluminación**

Figura 43.- Tipos de iluminación.

Puede notarse el efecto de la variación en la luminosidad en el entorno.

- Resolución. Es importante tener una misma resolución para todas las imágenes, ya que se obtendrán conjuntos de datos de dimensiones similares para su futura comparación.

La webcam, utilizada para el desarrollo de la aplicación se ajustó para capturar imágenes con formato y resolución: RGB24 / 120x160.

- Previa configuración de la webcam, para obtener condiciones de brillo, contraste, exposición, similares para cada captura.

Como ejemplo, se muestra la siguiente imagen la cual ha sido capturada con una previa configuración:

Figura 44.- Imagen con resolución de 120*160.

3.1.2 Conversión de la imagen a escala de grises.

Con el fin de tratar la menor cantidad posible de información en las imágenes, considerando además que las características que se buscan resaltar no requieren el procesado del color, se estableció el uso de las imágenes en formato de escala de grises.

Las imágenes son del mismo tamaño, 120x160 píxeles, esto debido a la resolución con la que se configuró la webcam. El procesado de la imagen inicia convirtiéndola de formato rgb a escala de grises. En las imágenes en formato rgb cada elemento de la matriz está constituido por un vector de tres números, cada uno asociado al valor de una de las componentes de color de dicho formato, en las imágenes en escala de grises la matriz se compone de elementos que indican el nivel de gris de cada píxel. Esto

significa que el tamaño de una imagen en escala de gris es mucho menor que el de una en rgb. Una imagen en rgb que posee un tamaño de 139x190x3 = 79230 Bytes en escala de gris sería solo de 139x190 = 26410 Bytes.

Un ejemplo sencillo de una conversión se realiza de la siguiente forma en Matlab:

>> imagen = imread(buskfoto);

Figura 45.- Ventana de diálogo para ver imagen a cargar.

Los comandos presentados en la Figura 45, especifican que se ha cargado en la variable `imagen` la foto de la seña “w”.

El tamaño de la variable imagen, es el siguiente:

>> size(imagen)

ans =

120 160 3

Lo anterior indica que la matriz tiene 120 filas y 160 columnas para cada componente de color (Rojo, Verde, Azul). En la Figura siguiente se observa la ventana de comando de windows, en la cual se visualiza el valor de cada celda de la imagen cargada.

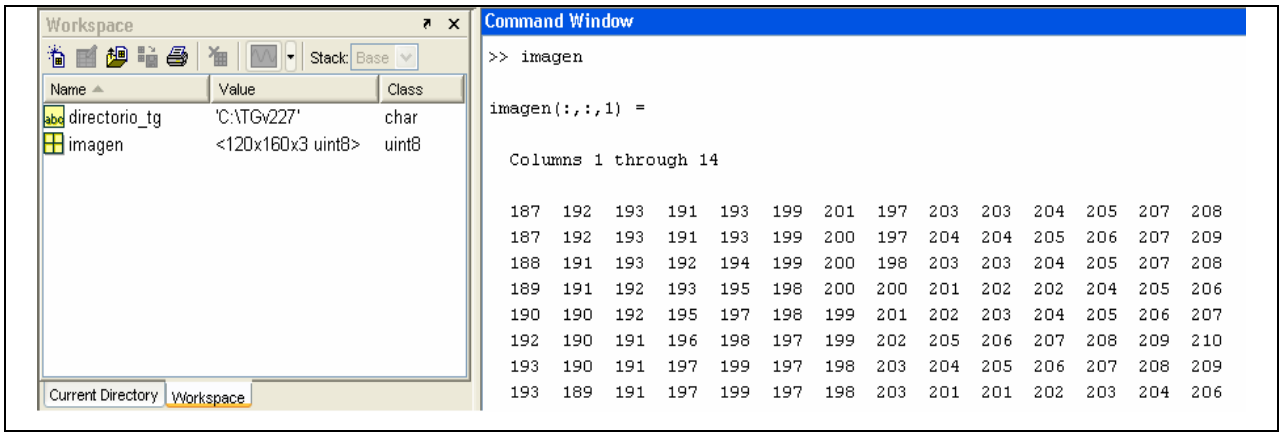


Figura 46.- Matriz de la imagen cargada.

En la Figura 47, se muestra la imagen JPG cargada con el comando imread.

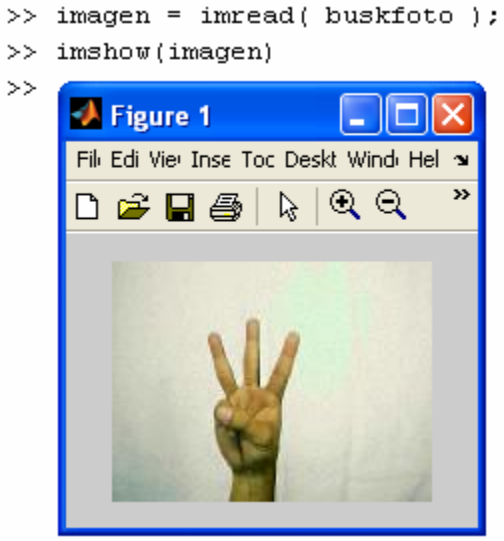


Figura 47.- Imagen cargada.

A esta variable se le aplica el comando `rgb2gray`, tal y como se muestra en la Figura 48, el cual la convierte a escala de grises.

```
>> imagen_rgb = rgb2gray (imagen);
>> size(imagen_rgb)

ans =

   120   160

>> imshow(imagen_rgb)
>>
```

Figura 48.- Muestra de la imagen transformada a escala de grises.

De este modo, se logra trabajar con una imagen que es 3 veces menor a la de color y se conservan las características de interés para el reconocimiento.

3.1.3 Detección de bordes.

Al obtener la imagen en escala de grises, se procede a realizar la extracción de características de la misma, las cuales representan una gran parte del procesamiento. Estas características son utilizadas en la simplificación de los datos que se incorporan posteriormente a las entradas de la red neuronal.

Para extraer dichas características, se han realizado varias pruebas con diferentes detectores de bordes, los resultados se describen a continuación:

3.1.3.1 Detector de bordes de sobel.

El primer detector de bordes utilizado fue una rutina de Sobel que consiste en una función en donde se especifica el nombre de la variable que contiene la imagen de entrada, el código es el siguiente:

```
function b=sobel(a)
% rutina para el detector de bordes de Sobel
% b=sobel(a)
% a: imagen de entrada.
% b: imagen de salida.

a=double(a);
w1=    [-1, 0, 1;
        -2, 0, 2;
       -1, 0, 1 ];

w2=    [ 1, 2, 1;
         0, 0, 0;
        -1,-2,-1];
c=abs(conv2(a,w1,'valid'));
cc=abs(conv2(a,w2,'valid'));
c=c+cc;
t1=c<=255;
t2=c>255;
c=c.*t1+255*t2;
b=uint8(c);
```

Los vectores de pesos w1 y w2 se describen como máscaras de gradientes, donde el gradiente define variaciones locales de la intensidad de la imagen.

En este caso w1 es la máscara utilizada para la detección de bordes verticales y w2 se utiliza para los horizontales.

El detector de bordes de Sobel es relativamente insensible al ruido, y ofrece un buen desempeño.

Pueden lograrse mejores características al ruido al usar vecindades más grandes a expensas de un mayor gasto de tiempo computacional. Sin embargo, vecindades mayores tienden a producir bordes gruesos.

La imagen de entrada tiene 120 x 160 píxeles y está en escala de grises, A esta imagen se le aplican las dos máscaras de gradiente de Sobel y los resultados que se obtienen son los siguientes:

		
(a) Imagen en RGB.	**(b) Imagen en escala de grises.**	**(c) Bordes de la imagen**

Figura 49.- Proceso de una imagen utilizando detector de bordes Sobel.

Estas imágenes siguen siendo del mismo tamaño de las que están en escala de grises, y los valores de sus matrices son variaciones entre 0 y 255 (negro y blanco respectivamente).

Las líneas de los bordes se han resaltado en la imagen, aunque en el interior de la mano se ven diferentes puntos blancos que representan ruido. Los bordes se ven

gruesos y la imagen no está binarizada. Para obtener una matriz binarizada (de 1 bit por píxel) es necesario aplicar otra rutina llamada “umborde”:

```
function b=umborde(a,p,nw,mw)
% Rutina para umbralizar los bordes detectados
% usando la media aritmética
% b=umborde(a,p,mw,nw)
% a: imagen de entrada.
% p: porcentaje que indica el nivel de umbralizado
% arriba de la media aritmética local
% mw: ancho de la ventana.
% nw: alto de la ventana.
% b: imagen de salida.

a=double(a);
w=ones(nw,mw)*(1+p)/(nw*mw);
t=ceil(conv2(a,w,'same'));
c=a>=t;
b=c.*255;
c=a<t;
b=uint8(b+c);
```

Mediante pruebas se determinaron los valores del porcentaje de nivel de umbralizado, ancho de ventana y alto de ventana, de 0.9, 25 y 25 respectivamente. El umborde con los valores mencionados, da como resultado las siguientes imágenes:

Figura 50.- Imagen binarizada.

La imagen binarizada solo posee dos colores blanco y negro; debido a la iluminación y a la rutina que se utilizó para la detección de bordes, se obtiene mucho ruido que se representa en los diferentes puntos blancos que rodean la mano, este ruido puede afectar el proceso de aprendizaje y el reconocimiento de la seña en la red neuronal. Es de mencionar que la imagen binarizada posee la cantidad de 118 x 158 pixeles, solo ha sido recortada dos filas y dos columnas de la original, esto indica que no se ha efectuado ningún proceso de compresión de los datos.

3.1.3.2 Función edge de Matlab.

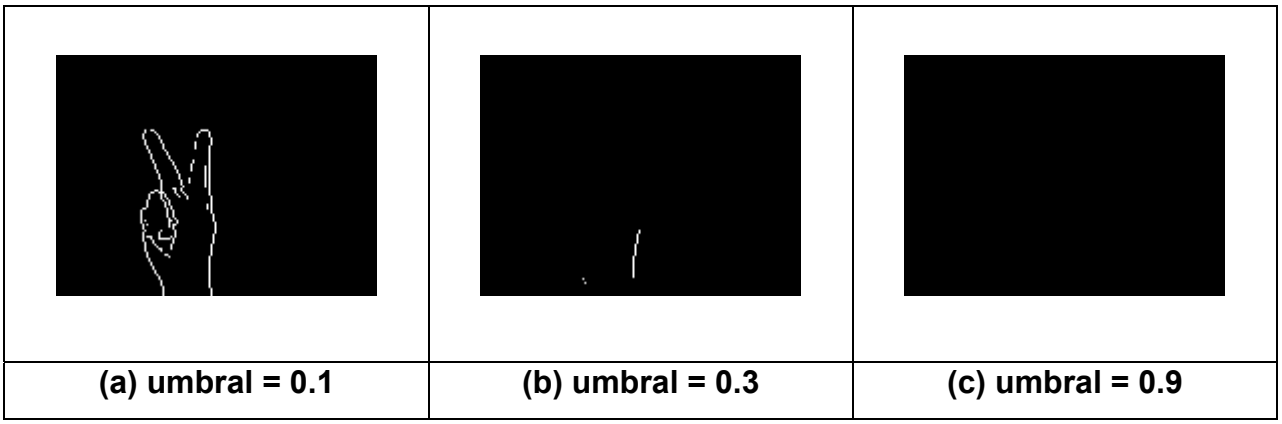
La función edge de Matlab, detecta los bordes. Esta función tiene diversos métodos en detección de bordes[5], los métodos que se utilizarán para realizar las pruebas son los siguientes:

- Sobel
- Prewitt
- Roberts
- Canny

- **SOBEL**

Este Sobel es mucho más fácil de utilizar que el anterior, como argumento de la función "edge" solo es necesario especificar el parámetro del umbral el cual ha sido determinado mediante pruebas según los siguientes parámetros:

bordes_de_imagen=edge(imagen_gris,'sobel',umbral)



(a) umbral = 0.1	(b) umbral = 0.3	(c) umbral = 0.9

Figura 51.- Detector de bordes Sobel con diferentes valores de umbral.

En la Figura 51 se observa que al aumentar el valor del umbral, los bordes se van perdiendo hasta que solo se tiene un fondo negro. La iluminación de la foto capturada también influye en la detección de bordes La imagen queda con bordes más finos que la rutina de Sobel utilizada anteriormente.

- **PREWITT**

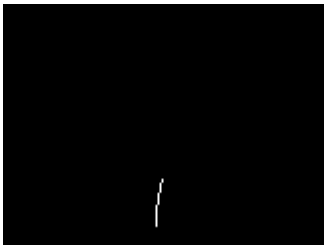
		
(a) umbral = 0.1.	**(b) umbral = 0.3.**	**(c) umbral = 0.9.**

Figura 52.- Detector de bordes Prewitt con diferentes valores de umbral.

Los resultados son bastante similares a los de Sobel, es decir, según se aumenta el umbral, la imagen se pierde.

- **ROBERTS**

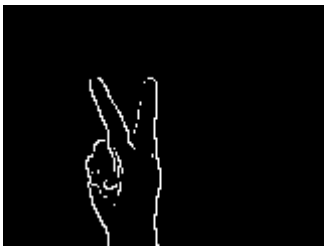
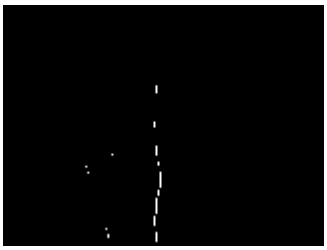
		
(a) umbral = 0.1.	**(b) umbral = 0.3.**	**(c) umbral = 0.9.**

Figura 53.- Detector de bordes Roberts con diferentes valores de umbral.

La Figura 53, presenta una ligera variación con umbral de 0.3 pero los resultados no cambian mucho con respecto a los anteriores.

- **CANNY**

		
(a) umbral = 0.1.	**(b) umbral = 0.3.**	**(c) umbral = 0.9.**

Figura 54.- Detector de bordes Canny con diferentes valores de umbral.

Observando los resultados de los diferentes métodos de detección de bordes, en la Figura 54 se comprueba que “canny” presenta muchas más líneas y detalles de la mano que los otros métodos. Por lo tanto se decide utilizar este detector de bordes en el procesamiento de la imagen, porque se definen con mayor claridad las líneas de las manos, se puede eliminar el ruido del fondo con un valor adecuado de umbral y la imagen está binarizada, lo que facilita el siguiente paso que consiste en la compresión de los datos.

3.1.4 Función bwmorph.

La aplicación del algoritmo bwmorph viene dado, con el propósito de mejorar la calidad de la imagen binarizada, el algoritmo se aplicará después de haber obtenido la imagen con el detector de bordes canny. El objetivo principal es el de encontrar un procedimiento que modifique la imagen de la mano (formato blanco y negro) con alguna característica visual mejorada, sin perder la información inicial de la imagen, esto con el propósito de mejorar la imagen antes de ser introducida a la red neuronal.

Para ver los resultados de varios procedimientos morfológicos de procesamiento digital de imágenes se usó el comando o función de matlab 'bwmorph'. La sintaxis de esta función es la siguiente:

(Obtenido de la ayuda en línea de Matlab: bwmorph (Image Processing Toolbox User's Guide)

imagen2 = bwmorph (imagen_BW , operation)

Este comando aplica una operación morfológica específica a una imagen en formato binario (blanco y negro).

imagen2 = bwmorph (imagen_BW , operation , n)

Con la instrucción anterior se aplica una operación 'n' veces, 'n' puede ser Infinito, en cuyo caso la operación se repite hasta que la imagen no sufra cambios.

Obteniéndose los siguientes resultados:

Tipo de imagen	Imagen a procesar	Resultado
Imagen de prueba		**(a)** Imagen capturada con la webcam
Imagen en escala de grises		**(b)** Imagen después de convertirla a escala de grises la imagen original
Imagen binarizada		**(c)** Imagen binaria obtenida de aplicar 'canny' a la imagen de escala de grises.

Figura 55.- Secuencia de una imagen capturada.

Con la imagen binarizada se aplicarán los siguientes métodos morfológicos:

Procedimiento morfológico	Imagen Resultante	Resultado
bwmorph(imc,'bothat')		**(a)** Solamente se resaltan los puntos donde se concentran muchos píxeles.
bwmorph(imc,'bridge')		**(b)** Agrega algunos puntos de ruido.
bwmorph(imc,'clean')		**(c)** No hay cambios.
bwmorph(imc,'close')		**(d)** Rellena algunos espacios cerrados con pixeles activos.

bwmorph(imc,'diag')		**(e)** Hace más gruesos los bordes.
bwmorph(imc,'dilate')		**(f)** Hace más gruesos los bordes, aún más que con el argumento 'diag'.
bwmorph(imc,'erode')		**(g)** Deja la imagen negra, solo el fondo.
bwmorph(imc,'fill')		**(h)** No hay cambios.
bwmorph(imc,'hbreak')		**(i)** No hay cambios.

bwmorph(imc,'majority')		**(j)** Solo quedan unos cuantos puntos blancos.
bwmorph(imc,'open')		**(k)** Solo queda el fondo negro.
bwmorph(imc,'remove')		**(l)** No hay cambios.
bwmorph(imc,'shrink')		**(m)** Elimina cierto ruido.
bwmorph(imc,'skel')		**(n)** No hay cambios.

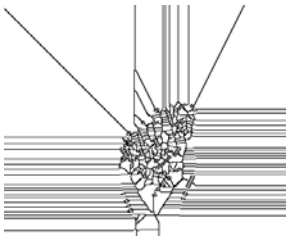
bwmorph(imc,'spur')		**(o)** No hay cambios.
bwmorph(imc,'thicken',Inf)		**(p)** Esqueleto con rayas.
bwmorph(imc,'thin',Inf)		**(q)** No hay cambios.
bwmorph(imc,'tophat')		**(r)** No hay cambios.

Figura 56.- Procedimientos morfológicos de una binarizada,

Según los resultados obtenidos con el bwmorph, se han tomado 3 procesos morfológicos, con el fin de mejorar la imagen binarizada. Estos algoritmos son:

- **Shrink:** Este algoritmo es aplicado para eliminar ruido presente en la imagen binarizada.

- **Dilate:** Con este algoritmo se obtienen bordes más gruesos, por lo que se puede observar una mejor imagen.

- **Close:** El algoritmo Close cierra algunos espacios cerrados con píxeles activos los cuales logran unir líneas abiertas que no fueron detectadas por el canny.

3.1.5 Compresión de la imagen.

Para poder entrenar la red neuronal, se necesita un vector columna en la entrada, pero con la menor cantidad de datos posibles para que el entrenamiento se realice de una forma rápida.
Los procesos de compresión de la imagen utilizados se describen a continuación:

3.1.5.1 Rutina MAX3.

Este programa selecciona bloques de 3x3 de la matriz de la imagen umbralizada, se hace una verificación del numero de ceros o unos que posee esa matriz, si el número de ceros es mayor, se coloca un cero en una celda de otra matriz sin datos, si el número se unos es mayor, se coloca un uno en la matriz vacía. De esta manera se va formando la matriz de la imagen comprimida.

Las imágenes se comprimen a la tercera parte de la original, por ejemplo si la imagen umbralizada tiene un tamaño de 137x188 = 25756 bytes, la imagen comprimida se reduce a un tamaño de 45x62 = 2790 bytes.

Esta rutina, toma pequeñas matrices de 3 x 3 píxeles de la imagen a esta matriz se le calcula su valor máximo, y éste es colocado en una matriz en blanco, la cual formará la nueva imagen. Esta compresión lleva muchas pérdidas la imagen es reducida a 1/3 de la original, por lo que se pierden muchos detalles. El código es el siguiente:

```
function mat_max = max3( mat )
 [ nf_1 , nc_2] = size( mat );
nf = (floor(nf_1/3))*3;
nc = (floor(nc_2/3))*3;
mat_max=[];
for cf = 1:3:nf ;
   for cc = 1:3:nc ;
   val_max = max( max( mat( cf:cf+2 , cc:cc+2 ) ));
   mat_max( floor( (cf/3) + 1) , floor( (cc/3) + 1) ) = val_max;
   end
end
mat_max=uint8(mat_max);
```

Las imágenes quedan de la siguiente forma:

 **(a)** Imagen de 120 x 160 pixeles en escala de grises.	 **(b)** Imagen de 40 x 53 pixeles en escala de grises (aplicandole max3).	 **(c)** Bordes de imagen con rutina sobel y umborde.
 **(d)** Imagen 40*53 píxeles (reducción de la imagen de bordes con max3).	 **(e)** Imagen depués de aplicar detector de bordes Canny.	 **(f)** Imagen reducida con la rutina max3.

Figura 57.- Aplicación del algoritmo Max3.

En la Figura 57(b), se observa que con la rutina max3 se pierden muchos datos importantes de la imagen original, la Figura 45 (d) es un ejemplo de la aplicación rutina Sobel siempre se tiene el ruido y la imagen ya no es muy diferenciable para el ojo humano. Con el detector de bordes Canny, ver Figura (f), la imagen se pierde completamente. Por esta razón no se utiliza la rutina max3 para la compresión de imágenes.

3.1.5.2 DCT (Transformada Discreta Coseno).

Una de las pruebas que se realizaron para reducir la imagen fue la aplicación de la transformada discreta del coseno. Ésta no es una idea original, ya que es un procedimiento utilizado en el algoritmo de compresión de imágenes del formato JPEG.
La dct de una imagen en escala de grises, se muestra en las siguientes Figuras.

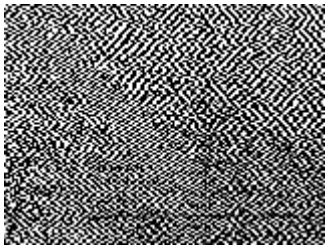
	
(a) Imagen en escala de grises.	**(b)** Tranformada discreta del coseno.

Figura 58.- Transformada discreta del coseno de una imagen en escala de grises

Esta transformada tiene la característica de concentrar la información en la esquina superior izquierda de la imagen.

La transformada discreta del coseno de una imagen a la que se le ha aplicado bordes se muestra en la siguiente Figura:

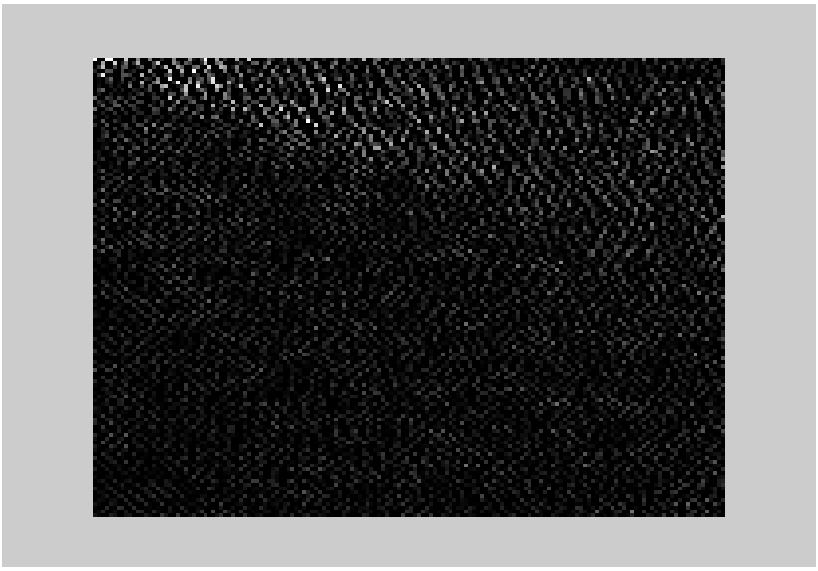


Figura 59.- Imagen de la transformada discreta del Coseno (DCT2)

Si se recorta la imagen a una cuarta parte de los datos, no se perderá la información importante, para ésto, será necesario convertirla a vector. Esta conversión también facilitará el entrenamiento de la red, ya que para realizarlo es necesario obtener un vector de entrada. La conversión a vector se realiza con la aplicación de un zigzag de la imagen.

El comando aplicado a este proceso es el siguiente:

```
foto_dct = dct2(foto);
vec_foto_dct = zigzag(foto_dct);
data_red = vec_foto_dct(1:1200);
```

La imagen foto es la proveniente del detector de bordes canny, con el comando dct2 se le aplica la transformada discreta del coseno bidimensional y ésta es guardada en la variable foto_dct.

3.1.5.3 Conversión de Matriz a vector.

El programa zigzag toma la transformada y va convirtiendo en vector haciendo un zigzag sobre la imagen, en el siguiente esquema se puede apreciar como se crea el vector.

Trayectoria general, seguida para la formación de un vector a partir de la matriz de datos, utilizando `zigzag`.

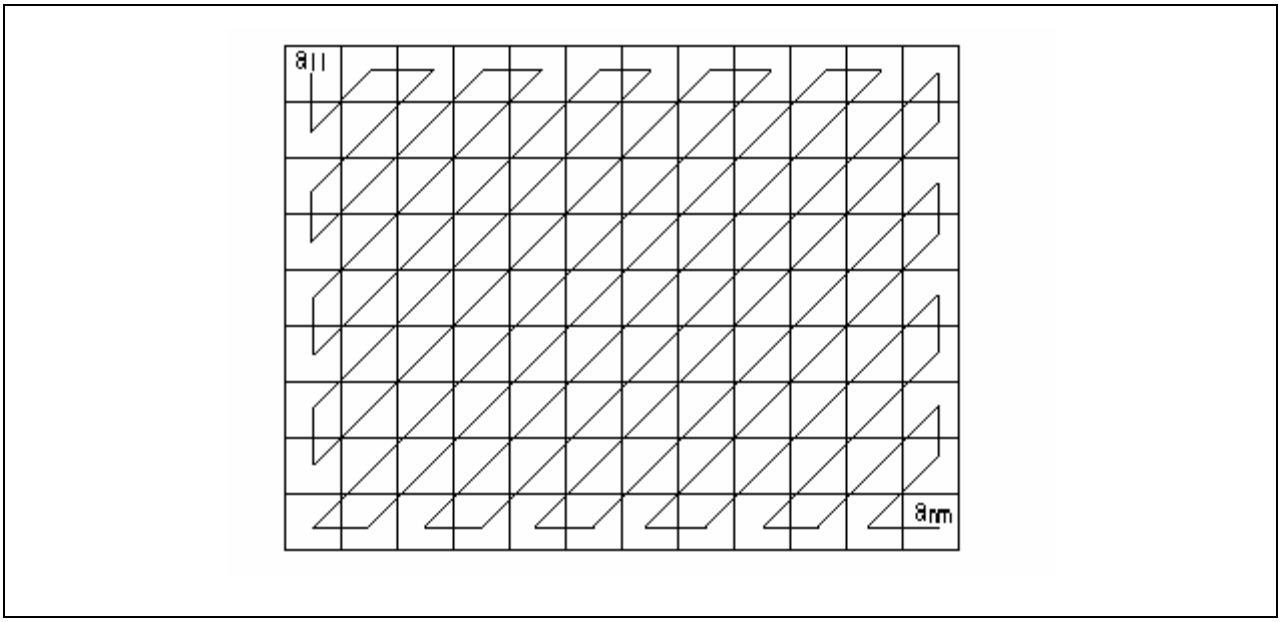
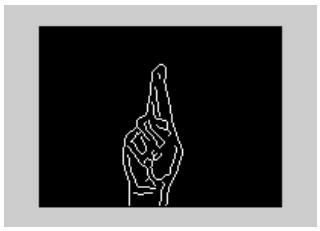


Figura 60.- Proceso de zigzag de una matriz

En la Figura 60 se puede observar la manera en que se van escogiendo los datos, comenzando por el píxel a_{11} seguido del a_{21}, a_{12}, a_{13}, a_{22}….a_{nm}.

Cuando se ha completado la rutina del zigzag, se recorta el vector a uno de (1,1200), con esto se reduce en gran manera la cantidad de datos. Para verificar si los datos no se pierden se necesita hacer un recorte de la matriz y aplicar la idct2 (proceso contrario a la dct2).



(a) Imagen resultante después del canny.

(b) Imagen recontruida después aplicar a DCT2.

Figura 61.- Imágenes antes y después de aplicar el Canny

La Figura 61 (a) es la imagen resultande del Canny, esta imagen es a la que se aplica la DCT2, la imagen reconstruida con la idct2 (Figura 61(b)) es de tamaño 120x160 con la diferencia de que está rellena de ceros, esto comprueba que se pueda recortar la imagen después de aplicar el zigzag, es decir no se perderá información. Es importante afirmar que los valores ya no están binarizados, a esto se debe que la imagen no quede totalmente clara.

Al entrenar la red, no se obtuvieron los resultados esperados, ya que no reconocía los datos de entrada, se analizaron las gráficas de amplitud contra posición de la dct2 y no eran parecidas, a continuación se presenta la gráfica de distintas letras “a”:

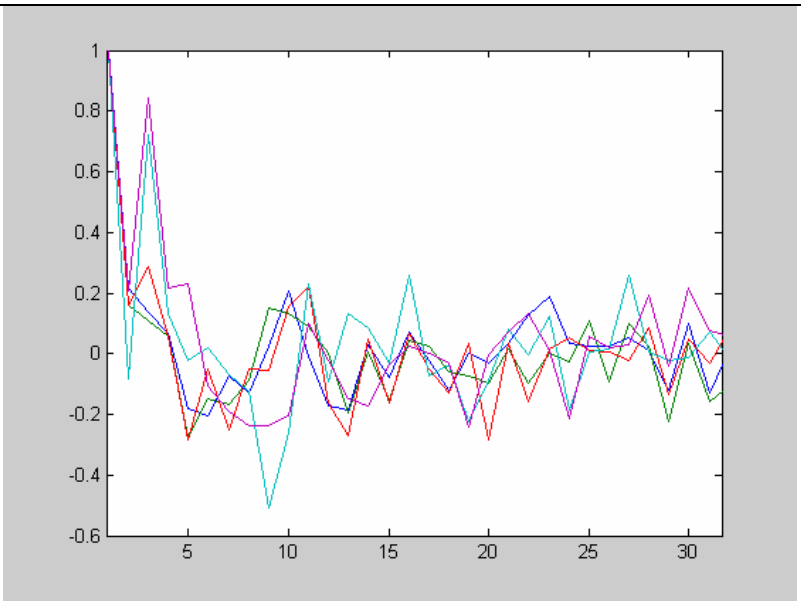


Figura 62.- Valor de DCT vrs Posición de mano

La gráfica muestra la no concordancia entre las mismas letras, esta es una de las razones por las cuales la red no reconoce, ya que a pesar de ser la misma letra pero con distinta mano no tienen la misma tendencia, por esta razón no se utilizará la transformada discreta del coseno.

3.1.5.4 Recorte de mano.

Se decidió realizar otro procedimiento para comprimir los datos de la imagen.
Una vez que se obtienen los bordes de la imagen con canny, se requiere reducir el tamaño de ésta, por lo tanto se aplica una rutina de recorte para dejar solamente la mano, logrando no solo reducir el tamaño sino que también no se pierde la información como se perdía con la DCT2.
Para que esta rutina de recorte trabaje con eficiencia, es necesario tener una buena iluminación y a la vez tener un fondo totalmente blanco, para evitar líneas en las partes inferiores de la imagen cuando se aplica el canny. A continuación se presentará el procedimiento completo de la imagen procesada.

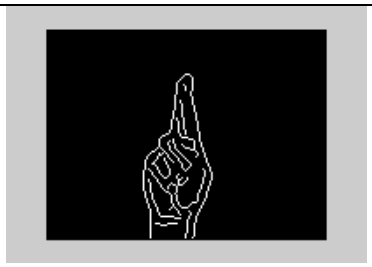
	
(a) Imagen original.	**(b) Escala de grises.**
	
(c) Aplicación de bordes (canny).	**(d) Imagen recortada.**

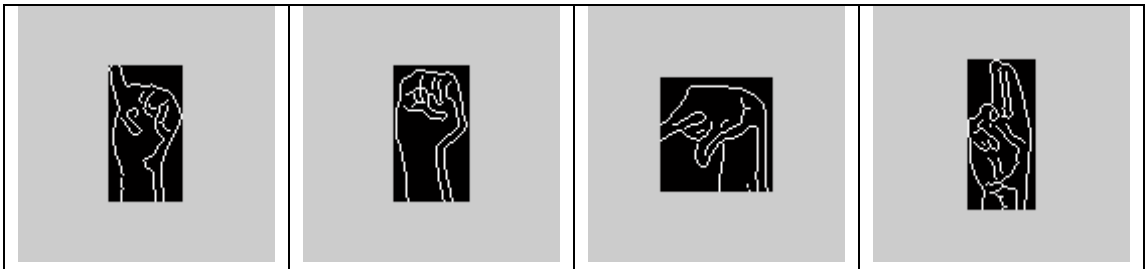
Figura 63.- Proceso de una imagen.

En la Figura 63, se describen los pasos utilizados para el procesamiento de la imagen, La Figura 63 (a), es la imagen digitalizada en RGB, La Figura 63(b), muestra la imagen en escala de grises por lo que solamente se obtiene una matriz de las tres primeras, La Figura 63(c), nos muestra la imagen resultante del la aplicación de bordes, utilizando el método Canny, para finalizar en la Figura 63(d) se puede observar el resultado de la rutina de recorte, en ella se presenta la información necesaria, ya que la parte del fondo se ha eliminado y solamente queda la mano, que es la información principal.

El algoritmo de recorte de la imagen es el siguiente:

```
function imnew = recorte(imold)
% Esta funcion encuentra los bordes de una imagen binarizada
% y reduce el tamaño de la misma hasta esos bordes
% imold debe ser la matriz correspondiente a la imagen binarizada
matg = imold;
[nf nc] = size(matg);
vecc = matg(:);
bord = find(vecc);
prims = bord(1:15);
lo = max(prims)/nf;
x1 = round(lo)-1;
ultis = bord(end-14:end);
le = min(ultis)/nf;
ancho = (round(le)+2)-x1+1;
matg = matg';
vecf = matg(:);
top = find(vecf);
ln = max(top(1:15))/nc;
y1 = round(ln)-3;
alto = (nf-y1)-5;
% Las coordenadas del rectángulo a recortar son:
```

```
% Esquina superior izquierda:
x1_y1 = [x1 y1];
% Esquina inferior derecha:
distancias = [ancho alto];
% Construyendo la imagen nueva:
imnew = imcrop(imold,[ x1_y1 , distancias ]);
```



(a) Letra i	(b) Letra e	(c) Letra q	(d) Letra u

Figura 64.- Imágenes utilizando el algoritmo de recorte.

Como ya se dijo anteriormente, para el entrenamiento de la red neuronal es necesario convertir las entradas de las imágenes a vector, estos vectores deben tener el mismo tamaño. Si se observa en la Figura 64, las imágenes recortadas no son iguales para todas, debido a la variación de la seña, estas pueden ser de diferentes formas y tamaño.

Estas variaciones hacen que el vector no sea de la misma longitud, por lo que se hace necesario elaborar una rutina que convierta todos los vectores a una misma cantidad de datos, para ello se estandarizará un vector de 2100x1; cuando se encuentren valores menores a esta dimensión, se agregarán ceros para llegar a la longitud deseada.

El algoritmo a utilizar es el siguiente:

```
vector = double(zigzag(foto_recortada));
data_red2 = zeros(2100,1);
[nf nc] = size(vector');
```

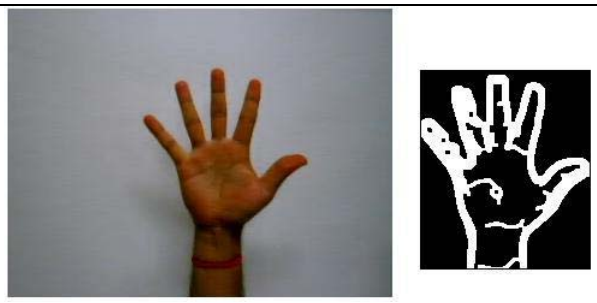
```
data_red2(1:nf,1:nc) = vector';
```

Después de obtener recorte se aplicará el zigzag antes mencionado para crear el vector y luego aplicar redondeo a 2100x1.

3.1.5.5 Mejoras al recorte de mano.

En las imágenes de la Figura 64 se observa que no se aplica recorte a la parte baja de la mano, es decir, el antebrazo algunas veces está y en otras no, por lo que se hace necesario desarrollar una rutina la cual no muestre la parte del antebrazo, con esto se obtendrán, menor cantidad de datos y a la vez la información que mas importante de la imagen. Para la implementación de esta rutina se colocará en la parte de la muñeca, una pulsera color negro, está servirá como referencia para saber hasta donde se realizará el recorte.

En al Figura 65(a) se observan señas, las cuales ya poseen la pulsera antes mencionada, en la Figura 65(b), se puede ver el resultado de la nueva rutina, es decir, la parte del antebrazo ya no aparece al hacer el recorte, por lo que se logra obtener la información necesaria para introducirla a la base de datos.



(a) Mano con pulsera

Figura 65.- Imágenes utilizando Pulsera.

3.2 Pruebas de reconocimiento para selección de Red.

Los tipos de reconocimiento utilizados en el desarrollo del trabajo se dividen en tres:

1. Eigvalues (valores propios).
2. Red Art.
3. Red Backpropagation.

PRUEBAS REALIZADAS PARA LA SELECCIÓN DE RED NEURONAL A UTILIZAR

RED NEURONAL ART 2

El entrenamiento de esta red se ha realizado cambiando diferentes parámetros, como los datos de entrada, el número de iteraciones, el parámetro de vigilancia, el bias y la razón de aprendizaje.

En la siguiente tabla se muestran las diferentes pruebas realizadas con las variaciones de lo parámetros de la red.

Número de entrenamiento	Cantidad de imágenes de entrada	Parámetro de vigilancia	Bias	Número de Iteraciones	Razón de Aprendizaje
1	26	0.75	0.000001	100	1
2	26	0.80	0.000001	100	1
3	26	0.77	0.000001	100	1
4	26	0.77	0.000001	400	0.95
5	26	0.80	0.000001	400	0.95
6	26	0.77	0.001	100	1
7	26	0.77	0.000001	400	0.97

Tabla 2. Valores utilizados en el entrenamiento de la red ART.

Entreno 1: El máximo número de categorías asignadas al entrenamiento es de 26, una categoría para cada letra: a b c d e f g h i j k l m n o p q r s t u v w x y z.

Se crearon 24 categorías, las letras G y H fueron asignadas a la misma categoría 7 y la letras B y W fueron asignadas a la categoría 2, para las otras letras se les asignó una categoría a cada una.

Se probó el reconocimiento con imágenes diferentes a las que se han utilizado para el entrenamiento, se tomaron 3 carpetas con 26 imágenes cada una:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	Reconoce	Reconoce	Reconoce
B	No reconoce	No reconoce	No reconoce
C	Reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	No reconoce	No reconoce	No reconoce
H	No reconoce	No reconoce	No reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	Reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	Reconoce
O	Reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	Reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	Reconoce	Reconoce	No reconoce

V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	Reconoce	Reconoce	Reconoce
Z	No reconoce	No reconoce	Reconoce

Tabla 3. Pruebas de reconocimiento, entreno 1.

Total de imágenes: 78

Total de reconocidas: 19

Porcentaje de efectividad: 24.36%

Entreno 2: Se crearon 26 categorías, una para cada letra que se está utilizando, en orden ascendente desde la A hasta la Z.

Se utilizaron las mismas tres carpetas para el reconocimiento de la imagen y se presentan los siguientes resultados:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	No reconoce	Reconoce
B	No reconoce	No reconoce	Reconoce
C	No reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	No reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	Reconoce	No reconoce	No reconoce
H	Reconoce	Reconoce	No reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce

O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	No reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	No reconoce	Reconoce	No reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 4. Pruebas de reconocimiento, entreno 2.

Total de imágenes: 78

Total de reconocidas: 10

Porcentaje de efectividad: 12.82%

Entreno 3: Al igual que el entreno 2 se crearon 26 categorías, una por cada letra. Los resultados del reconocimiento son los siguientes:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	Reconoce	Reconoce
B	Reconoce	No reconoce	Reconoce
C	Reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	Reconoce	No reconoce	No reconoce
H	Reconoce	Reconoce	Reconoce

I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	Reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	No reconoce	Reconoce	Reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 5. Pruebas de reconocimiento, entreno 3.

Total de imágenes: 78

Total de reconocidas: 17

Porcentaje de efectividad: 21.79%

Entreno 4: Se ha disminuido la razón de aprendizaje y se han aumentado las iteraciones. La cantidad de iteraciones necesarias para este entrenamiento ha sido de 253. Las categorías creadas son 26. Los resultados del reconocimiento se muestran en la siguiente tabla:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	Reconoce	Reconoce
B	No reconoce	No reconoce	No reconoce
C	Reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	Reconoce
G	No reconoce	No reconoce	No reconoce
H	Reconoce	Reconoce	Reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	Reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	No reconoce	No reconoce	No reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 6. Pruebas de reconocimiento, entreno 4.

Total de imágenes: 78

Total de reconocidas: 13

Porcentaje de efectividad: 16.67%

Entreno 5:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	No reconoce	Reconoce
B	No reconoce	No reconoce	Reconoce
C	No reconoce	No reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	No reconoce	No reconoce	No reconoce
H	Reconoce	Reconoce	No reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	No reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce

X	No reconoce	No reconoce	No reconoce
Y	No reconoce	Reconoce	No reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 7. Pruebas de reconocimiento, entreno 5.

Total de imágenes: 78

Total de reconocidas: 9

Porcentaje de efectividad: 11.54%

Entreno 6:

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	Reconoce	Reconoce
B	Reconoce	No reconoce	Reconoce
C	Reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	Reconoce	No reconoce	No reconoce
H	No reconoce	Reconoce	Reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce
M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	Reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce

T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	No reconoce	Reconoce	Reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 8. Pruebas de reconocimiento, entreno 6.

Total de imágenes: 78

Total de reconocidas: 16

Porcentaje de efectividad: 20.51%

Entreno 7:

Se necesitaron 100 iteraciones para realizar este entrenamiento.

Letra	Carpeta 1	Carpeta 2	Carpeta 3
A	No reconoce	Reconoce	Reconoce
B	Reconoce	No reconoce	Reconoce
C	Reconoce	Reconoce	No reconoce
D	No reconoce	No reconoce	No reconoce
E	Reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	No reconoce
G	No reconoce	No reconoce	No reconoce
H	No reconoce	No reconoce	No reconoce
I	No reconoce	No reconoce	No reconoce
J	No reconoce	No reconoce	No reconoce
K	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce

M	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce
R	No reconoce	Reconoce	No reconoce
S	No reconoce	No reconoce	No reconoce
T	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce
Y	No reconoce	Reconoce	Reconoce
Z	No reconoce	No reconoce	No reconoce

Tabla 9. pruebas de reconocimiento, entreno 7.

Total de imágenes: 78

Total de reconocidas: 13

Porcentaje de efectividad: 16.67%

Entreno 8: Se utilizará una carpeta diferente a las de los entrenamientos anteriores para este entrenamiento.

Total de imágenes: 78

Total de reconocidas: 13

Porcentaje de efectividad: 16.67%

A continuación se mostrarán las diferentes pruebas realizadas con los diferentes métodos,

Las pruebas del reconocimiento que se muestran a continuación son realizadas con la imágenes en las cuales se utiliza la pulsera.

CARPETAS	LETRA ERRONEA	LETRA RECONOCIDA	PORCENTAJE DE EFECTIVIDAD
FON	f	Q	**72%**
	g	P	
	c	L	
	m	T	
	p	H	
	r	U	
	v	i	
FON_1	c	o	**80%**
	m	t	
	p	c	
	q	z	
	w	k	
FON_2	b	i	**64%**
	c	o	
	f	q	
	i	f	
	m	s	
	p	c	
	u	r	
	y	f	
	z	f	

CARPETAS	LETRA ERRONEA	LETRA RECONOCIDA	PORCENTAJE DE EFECTIVIDAD
JUAN	d	i	**60%**
	e	i	
	l	p	
	n	m	
	o	c	
	p	q	
	q	p	
	r	d	
	s	n	
	x	q	
JUAN_1	b	u	**72%**
	h	p	
	n	a	
	p	q	
	r	u	
	t	m	
	v	p	
JUAN_2	b	r	**52%**
	c	d	
	g	h	
	k	v	
	l	q	
	m	a	
	n	a	
	p	q	
	r	v	
	t	n	
	u	k	
	v	k	

CARPETAS	LETRA ERRONEA	LETRA RECONOCIDA	PORCENTAJE DE EFECTIVIDAD
MARIO	c	o	**60%**
	h	g	
	k	r	
	m	n	
	p	h	
	t	s	
	u	r	
	v	u	
	w	u	
	z	c	
MARIO_1	c	f	**60%**
	g	h	
	i	w	
	k	d	
	m	n	
	n	t	
	r	d	
	u	d	
	w	x	
	z	f	
MARIO_2	c	o	**76%**
	g	h	
	k	v	
	m	n	
	n	t	
	r	v	
WENDY	c	l	**64%**
	h	p	
	k	d	
	l	f	
	n	m	
	o	i	
	q	p	
	r	u	
	v	k	

Tabla 10. Pruebas de reconocimiento, entreno 8.

Las tablas anteriores muestran el porcentaje de reconocimiento, es importante tomar en cuenta que los resultados obtenidos son pruebas hechas de cada carpeta de entrenamiento. El porcentaje total de reconocimiento es del 66%.

3.2.1 ROLES

Reconocimiento Óptico del Alfabeto del Lenguaje de Sordos

3.2.1.1PROCESOS

- El proceso aplicado a cada imagen adquirida es el siguiente:
- Captura de Foto con webcam.
- Conversión de Foto a Matriz (120x160x3).
- Foto (Matriz) en color a escala de grises.
- Redimensionado de Matriz (Foto) para primera reducción de datos.
- Realce de bordes con Canny.
- Redimensionado de imagen por eliminación de áreas vacías (Recorte).
- Conversión de matriz a vector columna.
- Reconocimiento.

3.2.1.2 ENTRENAMIENTO

Red Backpropagation.

Las primeras pruebas realizadas con la red neuronal backpropagation se dividen en tres, La primera red es para reconocer que tipo de clasificación es, cerrada o abierta, la categoría es tomada según sea la forma de la seña, la mano cerrada es aquella que no posee ningún dedo despegado del puño, la mano abierta tiene que tener por lo menos un dedo despegado del puño.

La red2, entrenara los datos solo de mano cerrada, es decir, que solo se tendrán 8 tipos de señas, 'a' 'c' 'e' 'm' 'n' 'o' 's' 't' (ver anexo), En la red3 se entrenaran los datos de la categoría de mano abierta, 'b' 'd' 'f' 'g' 'h' 'i' 'k' 'l' 'p' 'q' 'r' 'u' 'v' 'w' 'x' 'y' .

Debido a que se forma una división de las señas, la cantidad de datos será menor y el reconocimiento deberá tener un mejor porcentaje de error ya que si lo comparamos con

un entrenamiento de mayor cantidad de datos de entrada el porcentaje de error aumentará.

Los tipos de clasificación se dividen de las siguientes categorías:
a) Mano cerrada: { 'a' 'c' 'e' 'm' 'n' 'o' 's' 't' }
b) Mano abierta: { 'b' 'd' 'f' 'g' 'h' 'i' 'k' 'l' 'p' 'q' 'r' 'u' 'v' 'w' 'x' 'y' }

Red1 (Red de categorización)

El tipo red utilizada es una Backpropagation (red de retropropagación), posee una cantidad de entradas de 1200 con 3 capas, una de entrada, una oculta y una de salida.
El tipo de entrenamiento utilizado para la red backpropagation es Traingdx, la cual realizo 674 épocas de 50000 máximas, se obtuvo una sumatoria de error cuadrático de 0.000877594 de un 0.001 estimado, el gradiente fue de 0.00634306, el tiempo total de entrenamiento de esta red es de 4595 segundos, es decir, 1 h, 16 min, 35 seg.

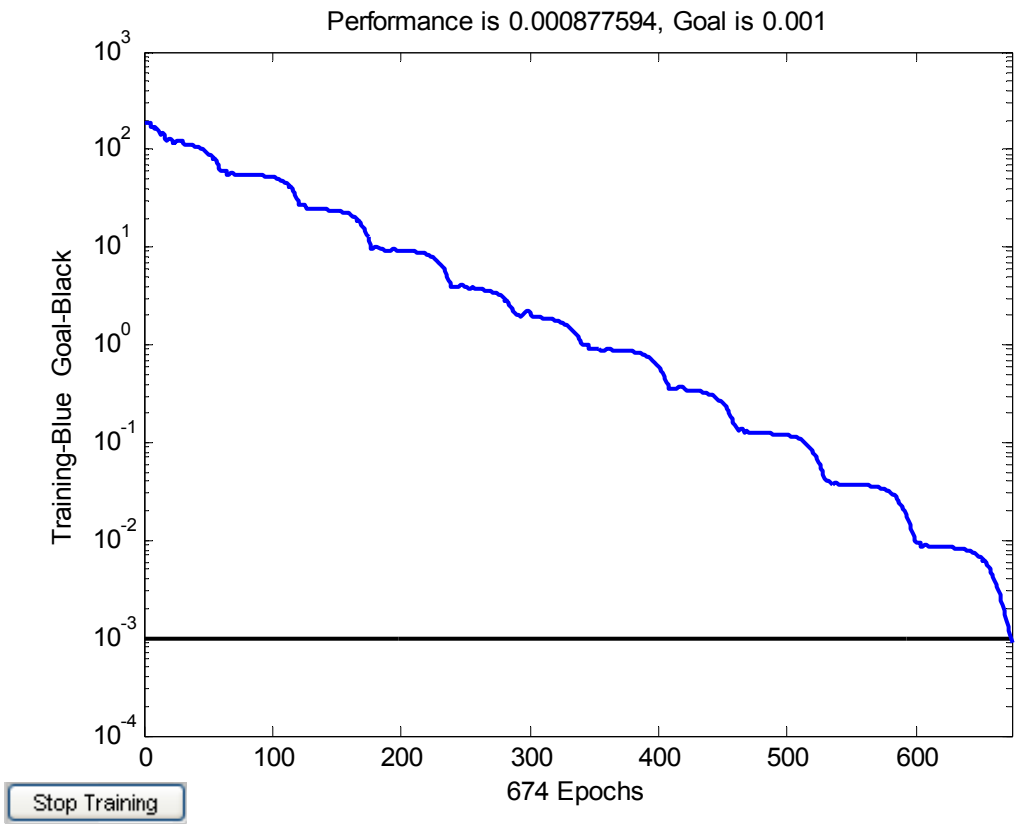


Figura 66.- Gráfica obtenida en el entrenamiento de red1.

TRAINGDX, Epoch 674/50000, SSE 0.000877594/0.001, Gradient 0.00634306/1e-020
Elapsed time is 4595 seconds. (1 hora, 16 min, 35 seg).

Red2 (Entrenamiento de mano cerrada)

Se utilizó una red backpropagation, el tipo de entrenamiento es el TRAINGDX, con 18902 iteraciones, y un error cuadrado medio de 0.00998319. Un gradiente de 0.705935, teniendo un tiempo de 59816 segundos, es decir, 16 horas, 36 min, 56 seg).

Red2, Mano cerrada (puño): { 'a' 'c' 'e' 'm' 'n' 'o' 's' 't' }

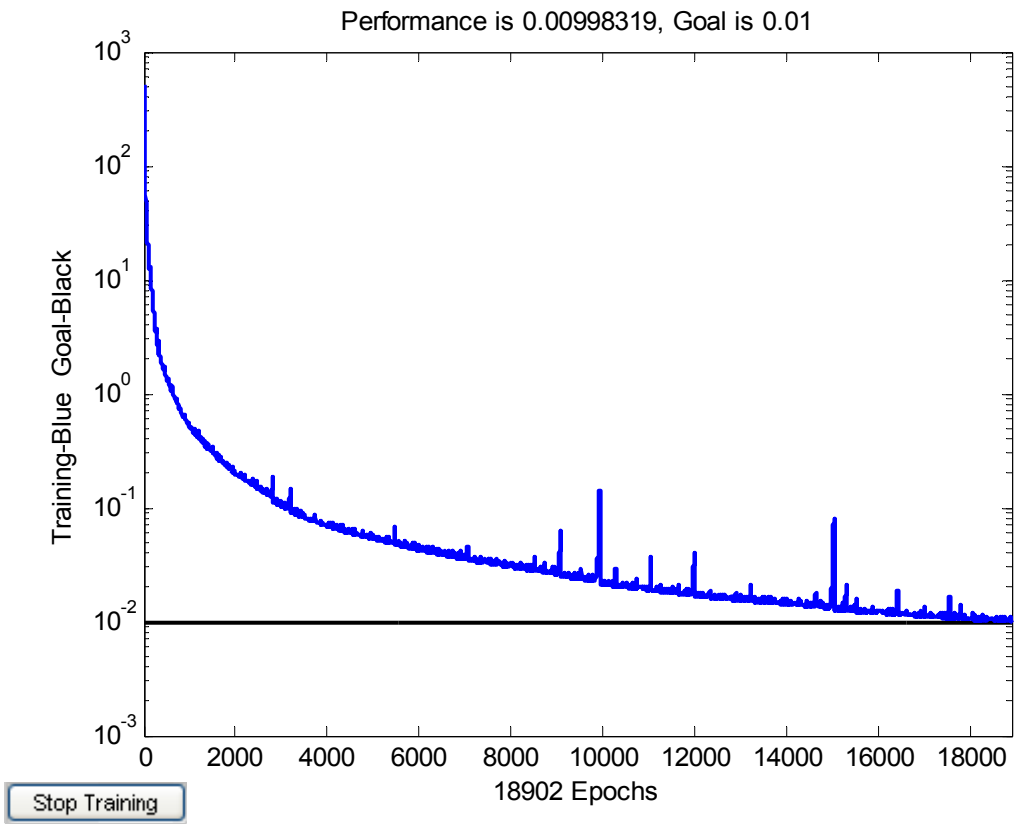


Figura 67.- Gráfica obtenida en el entrenamiento de red2.

Entrenamiento de red 2. Para el reconocimiento de las señas que se realizan con la mano cerrada (puño).

Resultados del entrenamiento:

TRAINGDX, Iteraciones: 18902/50000, Error Cuadrado Medio: 0.00998319/0.01, Gradiente: 0.705935/1e-020

Tiempo finalizado en: 59816 seconds (16 horas, 36 min, 56 seg).

Red3 (Mano abierta)

La red utilizada para este reconocimiento una backpropagation con un entrenamiento de tipo TRAINGDX, teniendo 33100 épocas, con un error cuadrado medio de 0.0348198 y un gradiente de 8.22272. El tiempo que tardó enl entrenamiento es de 15100 segundos, esdecir, 41 horas, 56 min, 41 seg.

Red3, Mano abierta: { 'b' 'd' 'f' 'g' 'h' 'i' 'k' 'l' 'p' 'q' 'r' 'u' 'v' 'w' 'x' 'y' }

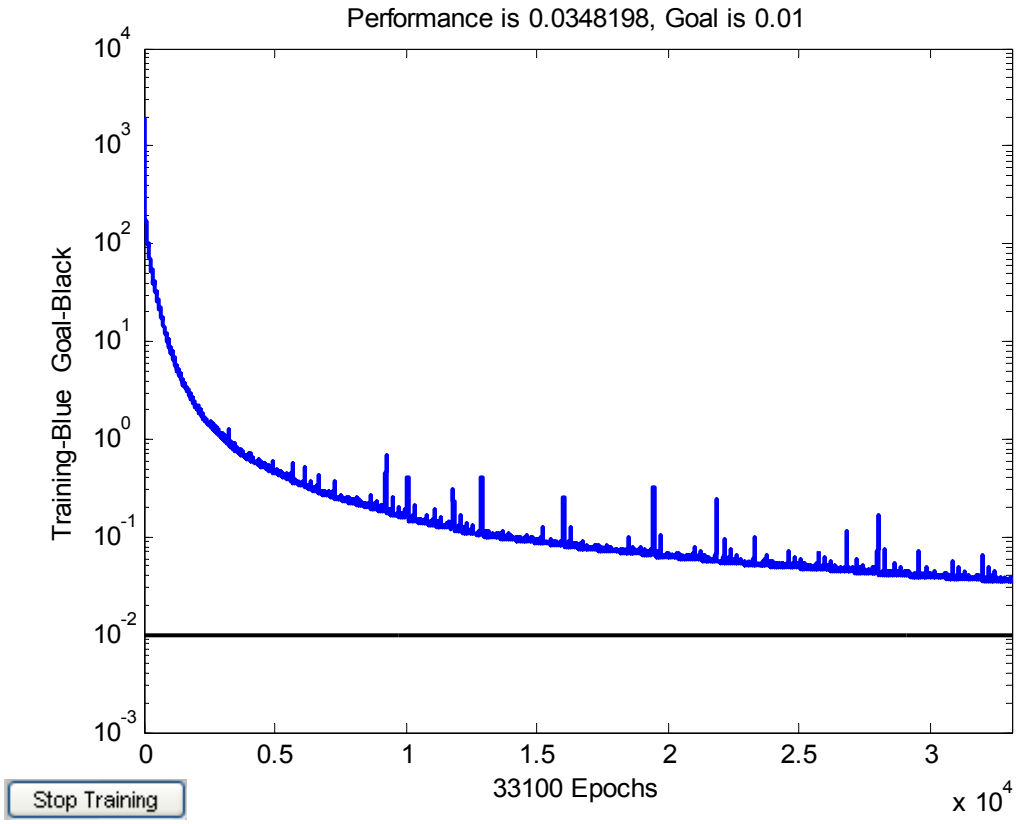


Figura 68.- Gráfica obtenida en el entrenamiento de red3.

Entrenamiento red 3 – Para reconocimiento de las señas que se realizan con al menos un dedo extendido (palma).

Entrenamientos Realizados para la versión final

RED NEURONAL ARTIFICIAL BACKPROPAGATION

Para implementar este entrenamiento, se utilizó una computadora con las siguientes características fundamentales:

Procesador: Intel Pentium IV (Hyper Treading) a 3GHz

Memoria RAM: 1GB

De implementarse en una pc con menor capacidad, los tiempos obtenidos se incrementan considerablemente.

La red neuronal (backpropagation) dentro del entorno de trabajo de Matlab, se define de la siguiente forma:

```
red =

    Neural Network object:

    architecture:

        numInputs: 1
        numLayers: 4
      biasConnect: [1; 1; 1; 1]
     inputConnect: [1; 0; 0; 0]
     layerConnect: [4x4 boolean]
    outputConnect: [0 0 0 1]
    targetConnect: [0 0 0 1]

       numOutputs: 1  (read-only)
       numTargets: 1  (read-only)
    numInputDelays: 0  (read-only)
    numLayerDelays: 0  (read-only)
```

```
subobject structures:

        inputs: {1x1 cell} of inputs
        layers: {4x1 cell} of layers
       outputs: {1x4 cell} containing 1 output
       targets: {1x4 cell} containing 1 target
        biases: {4x1 cell} containing 4 biases
  inputWeights: {4x1 cell} containing 1 input weight
  layerWeights: {4x4 cell} containing 3 layer weights

functions:

      adaptFcn: 'trains'
       initFcn: 'initlay'
    performFcn: 'sse'
      trainFcn: 'traingdx'

parameters:

    adaptParam: .passes
     initParam: (none)
  performParam: (none)
    trainParam: .epochs, .goal, .lr, .lr_dec,
                .lr_inc, .max_fail, .max_perf_inc, .mc,
                .min_grad, .show, .time

weight and bias values:

            IW: {4x1 cell} containing 1 input weight matrix
            LW: {4x4 cell} containing 3 layer weight matrices
```

b: {4x1 cell} containing 4 bias vectors

other:

userdata: (user stuff)

Una vez creada la variable que contiene las especificaciones sobre la arquitectura de la red y los parámetros utilizados para su entrenamiento, se procede a calcular los pesos para los cuales se espera obtener el error deseado en la salida de la red.

```
[R,Q] = size(P);
[S,Q] = size(T);
```

Donde:

P es la matriz de entrada, donde cada vector columna representa una imagen.

T es la matriz de salidas deseadas, en la que para cada vector columna, solamente 1 de las salidas debe activarse para identificar una imagen.

```
red2 = newff(minmax(P),[R 200 100 S],{'logsig','logsig','tansig','tansig'},'traingdx');

[red, tr] = train(red2,P,T);
```

El diagrama general de la red neuronal, según la notación utilizada en Matlab, es el mostrado en la Figura 69:

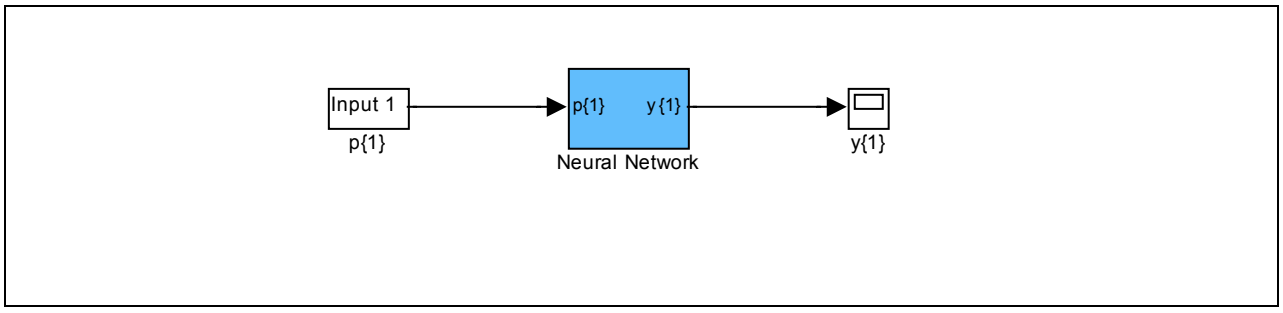


Figura 69 – Arquitectura de la Red Backpropagation entrenada.

Para generar el diagrama correspondiente a la red neuronal, puede escribirse en la línea de comandos de Matlab, lo siguiente:

>> load red;

>> gensim(red)

Para obtener las figuras que describen la arquitectura de cada componente de la red, basta con hacer doble clic en el bloque que lo representa en la ventana de simulik que se abrió al general la simulación de la red (gensim).

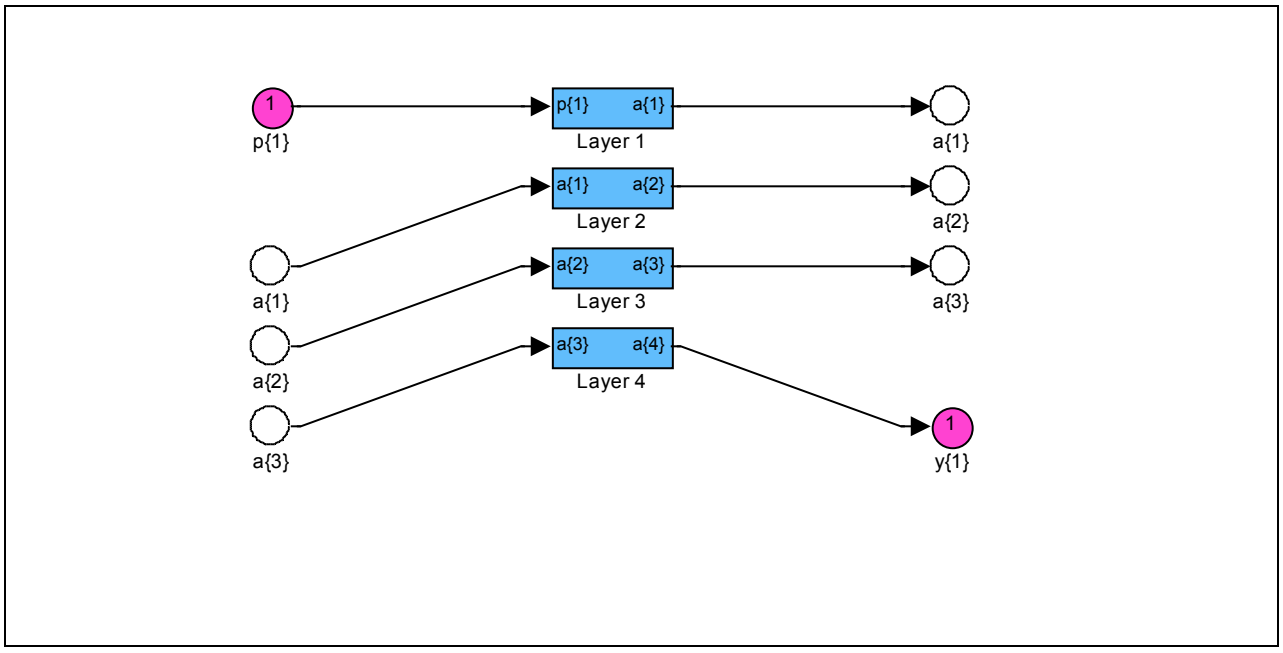


Figura 70 – Arquitectura que muestra la interconexión entre las capas.

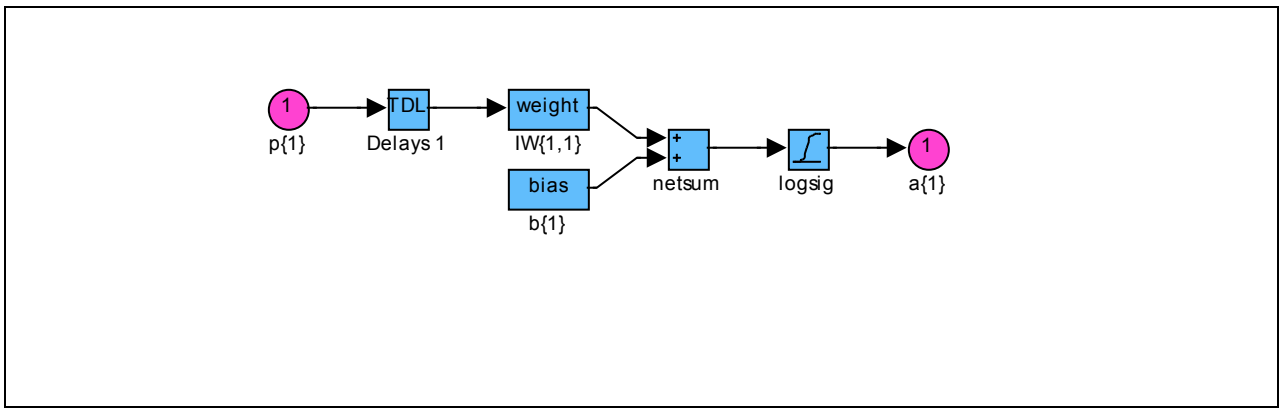
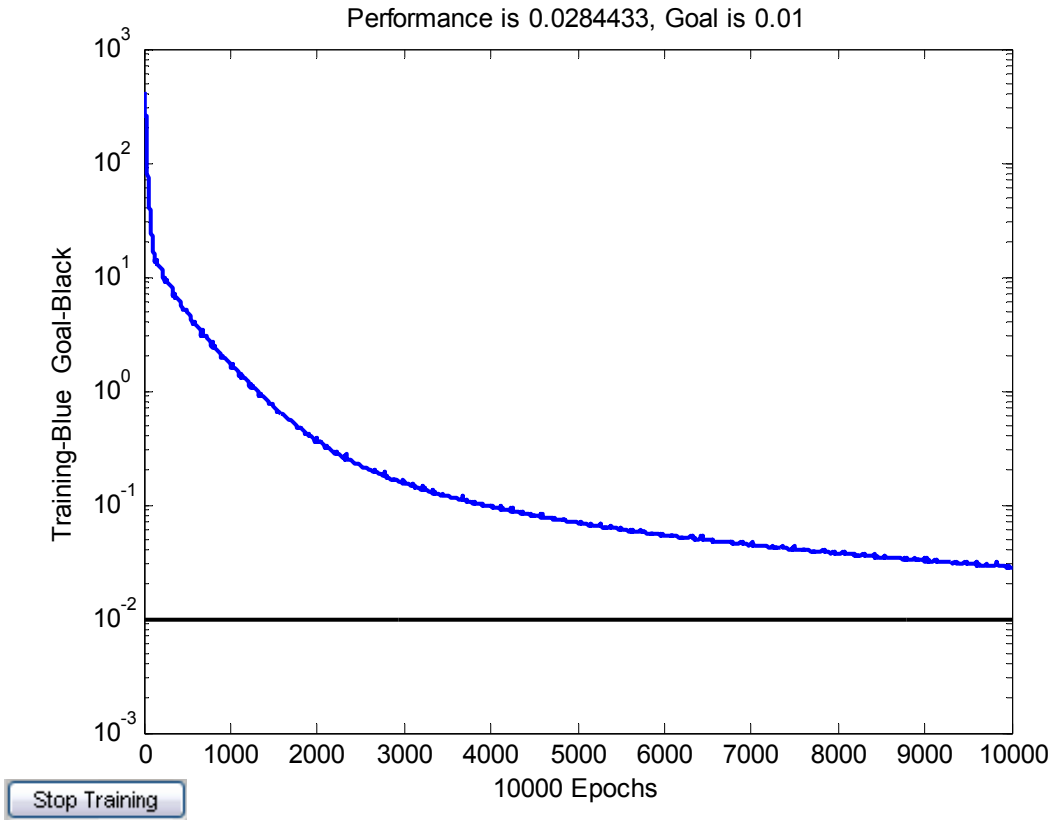


Figura 71 –Estructura elemental que compone cada una de las capas.

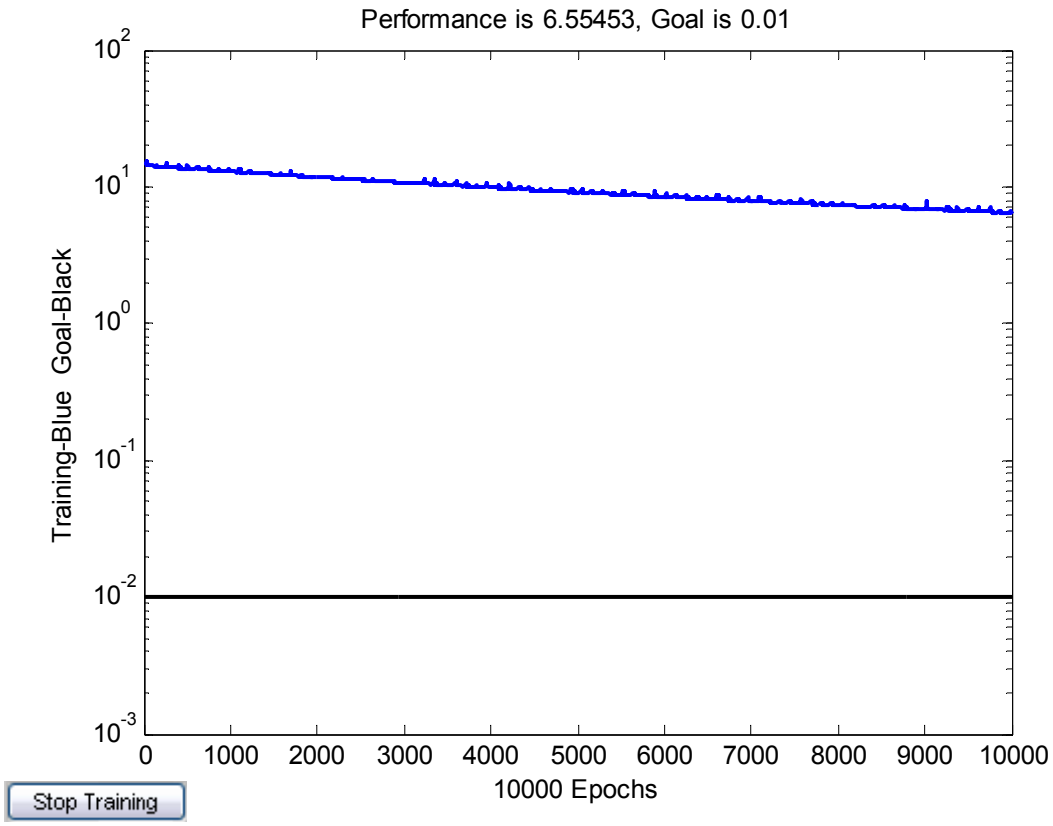
Entrenamiento del 2005.08.12:1015

Entrenamiento con un solo vector de entrada por cada imagen [p_E , p_N , p_W] con los perfiles obtenidos para una carpeta de imágenes ‘a01’



TRAINGDX, Epoch 10000/10000, SSE 0.0284433/0.01, Gradient 0.105991/1e-020
TRAINGDX, Maximum epoch reached, performance goal was not met.
Tiempo utilizado para el entrenamiento: menos de 20 minutos.

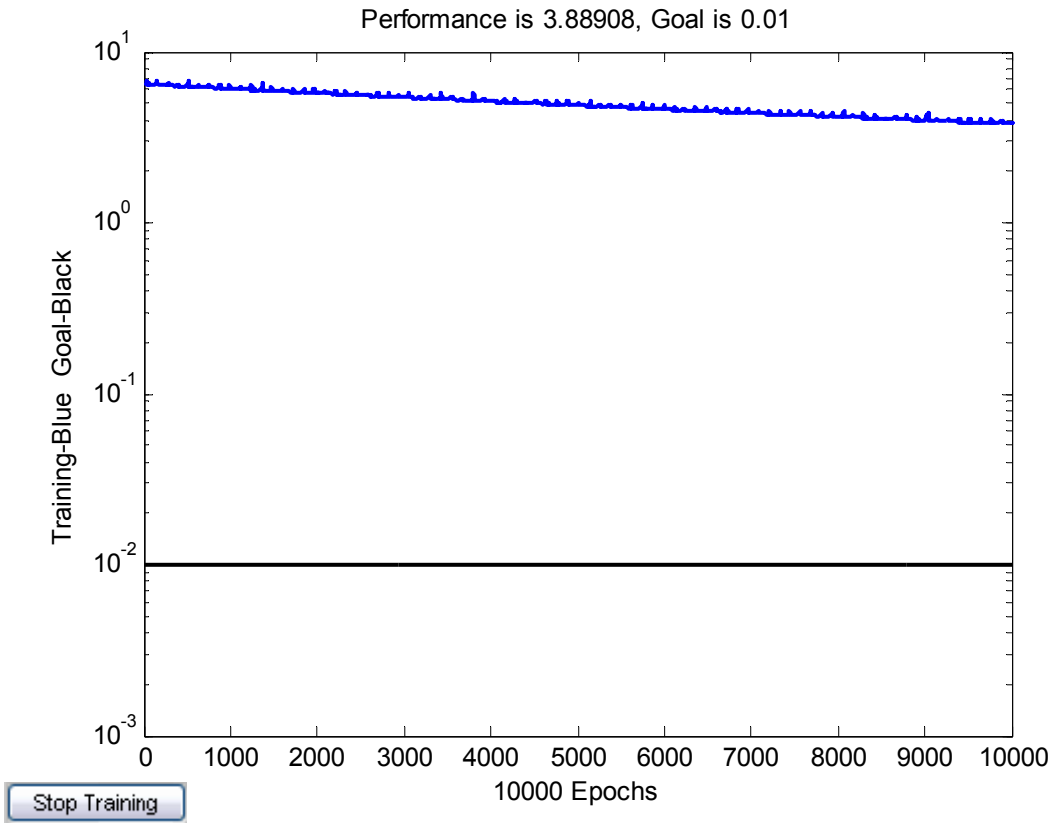
2005.08.12:1035
Ya que el entrenamiento con una sola carpeta de imágenes resultó satisfactorio, se procedió a realizar el entrenamiento con 6 carpetas de fotos diferentes, para tener un rango de variación mayo.
La red utilizada es la misma que para el entrenamiento 20050812:1015. Se usaron como pesos iniciales, los últimos que se calcularon en el entrenamiento mencionado.



TRAINGDX, Epoch 10000/10000, SSE 6.55453/0.01, Gradient 133.385/1e-020
TRAINGDX, Maximum epoch reached, performance goal was not met.

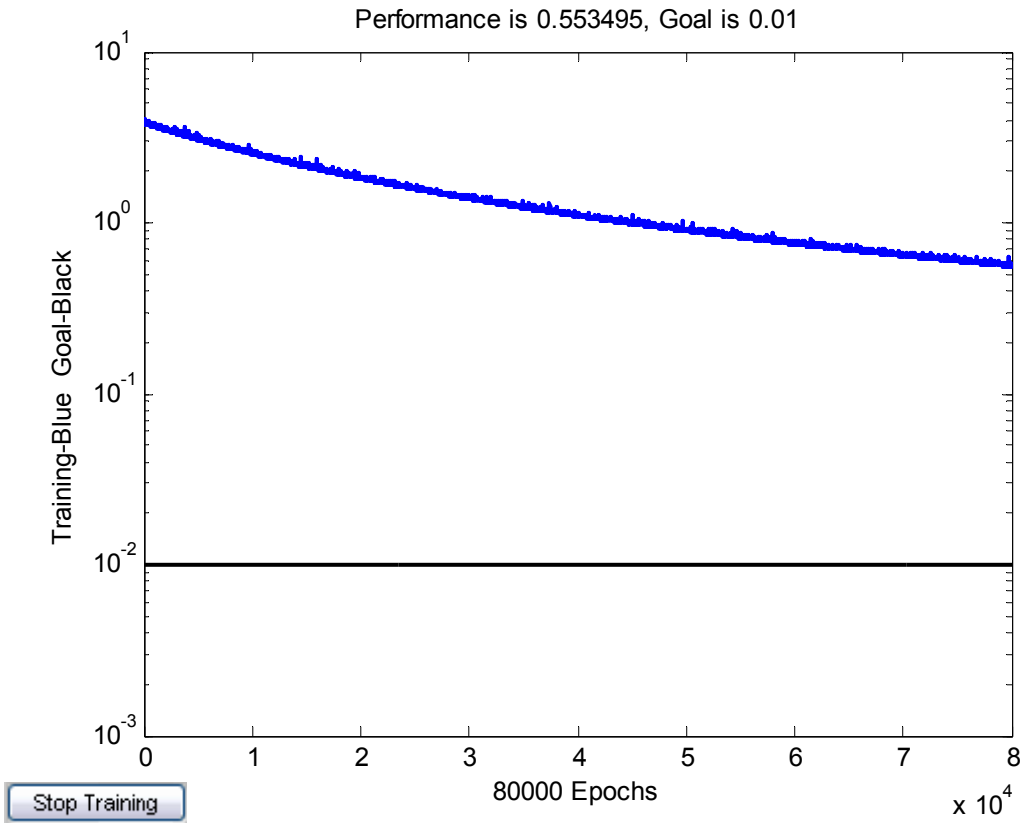
Tiempo demorado: aproximadamente 40 minutos

Segundo intento, tomando los últimos pesos calculados como pesos iniciales. Las entradas son las mismas, al igual que los vectores de salida.



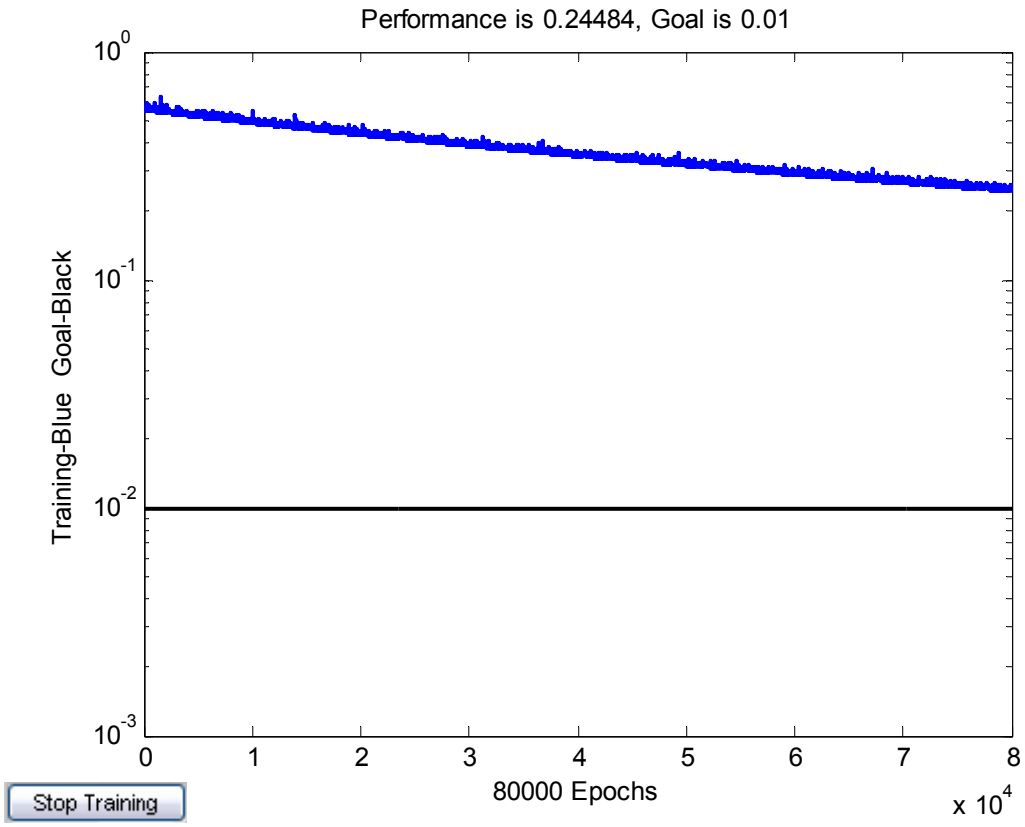
TRAINGDX, Epoch 10000/10000, SSE 3.88908/0.01, Gradient 142.591/1e-020
TRAINGDX, Maximum epoch reached, performance goal was not met.

Continuación del entrenamiento...



TRAINGDX, Epoch 80000/80000, SSE 0.553495/0.01, Gradient 0.646554/1e-020
TRAINGDX, Maximum epoch reached, performance goal was not met.

Se realizó el ultimo entrenamiento con la misma RNA, obteniendo los siguientes resultados:



TRAINGDX, Epoch 80000/80000, SSE 0.24484/0.01, Gradient 13.2281/1e-020
TRAINGDX, Maximum epoch reached, performance goal was not met.

Aunque no se alcanzó el sse esperado, se considera que el error alcanzado es suficiente para la función de categorización de las señas de manos, por lo que no se realizó más iteraciones.

Tiempo requerido para el entrenamiento: aproximadamente 5 horas.

El siguiente paso es la prueba de la red neuronal con nuevas fotos capturadas en tiempo real y en condiciones normales.

RED NEURONAL ARTIFICIAL ART

El proceso que consiste en el entrenamiento utilizado para la red ART se puede definir de una forma general por los siguientes pasos:

a) Crear una matriz que contiene los perfiles izquierdo, derecho y de arriba de la imagen procesada. Donde cada columna representa al vector de una letra del alfabeto.
b) Hacer un complemento de los datos de cada fila de la matriz, como resultado se obtiene una matriz con el mismo número de columnas pero con el doble de filas.
c) Crear la red neuronal utilizando como entrada la matriz de datos complementados.
d) Entrenar la red creada y presentar las categorías.

Para crear la matriz de datos de entrada, se necesita cargar las dimensiones siguientes:

a_altura : Constante de la razón de altura para la letra 'a' respecto al ancho de la muñeca.

b_altura : Constante de la razón de altura para la letra 'b' respecto al ancho de la muñeca.

c_luz : Constante para compensar el umbral para la binarización de la imagen de entrada.

Estas constantes son utilizadas por el programa proc_foto que realiza el procesamiento de la imagen de la siguiente forma:

[mano, mano_skel, p_A, p_I, p_D] = proc_foto(foto_mano, a_altura, b_altura,c_luz)

La función anterior devuelve los perfiles:

p_A : Vector de distancias hacia la mano desde arriba.

p_I : Vector de distancias hacia la mano desde la izquierda.

p_D : Vector de distancias hacia la mano desde la derecha.

Estos perfiles son almacenados en forma de vector en la matriz mat_manos que se utiliza como la entrada de la red neuronal.

Antes de crear la red se efectúa el complemento de los datos de la forma 1-x donde los datos originales representan la variable x. Por ejemplo, si se tienen los datos siguientes:

0.3 0.2 0.6
0.4 0.1 0.8

El complemento de datos dará como resultado:

0.3 0.2 0.6
0.7 0.8 0.4
0.4 0.1 0.8
0.6 0.9 0.2

La matriz a la que se ha aplicado el complemento es la nueva entrada de la red, por lo tanto se procede a crearla con la función:

Red_ART = ART_Create_Network(numFeatures)

Una estructura es creada tal y como se muestra en la figura, en donde:
numFeatures : Cantidad de filas de la matriz de entrada.
numCategories: El número de categorías que son creadas durante el entrenamiento.
maxNumCategories: El máximo número de categorías que pueden crearse en la red.
Weight : La matriz de pesos de la red.
Vigilance: El parámetro de vigilancia que se ha designado para entrenar la red.
Bias: Presenta su valor actual.
numEpochs: Cantidad de iteraciones de la red,
learningRate: Razón de aprendizaje, el valor de 1 indica el aprendizaje más rápido.

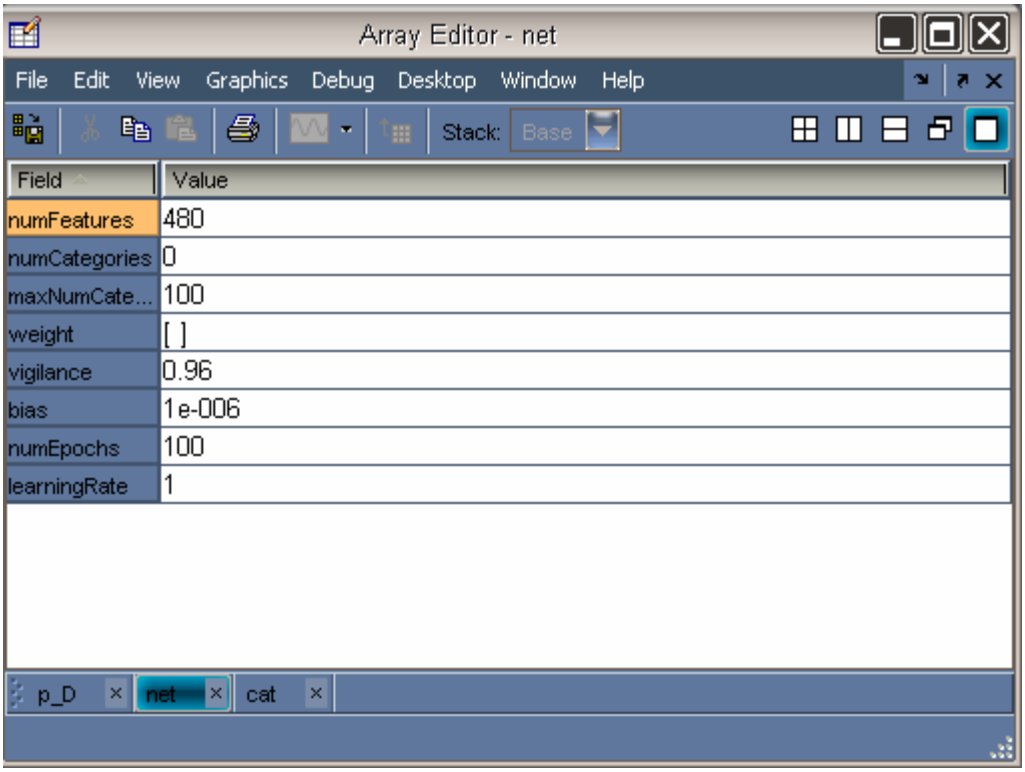


Figura 72 - Estructura de la red creada con ArtNetwork

Cuando la red ha sido creada se procede al entrenamiento, éste da como resultado una nueva estructura y el vector de las categorías creadas:

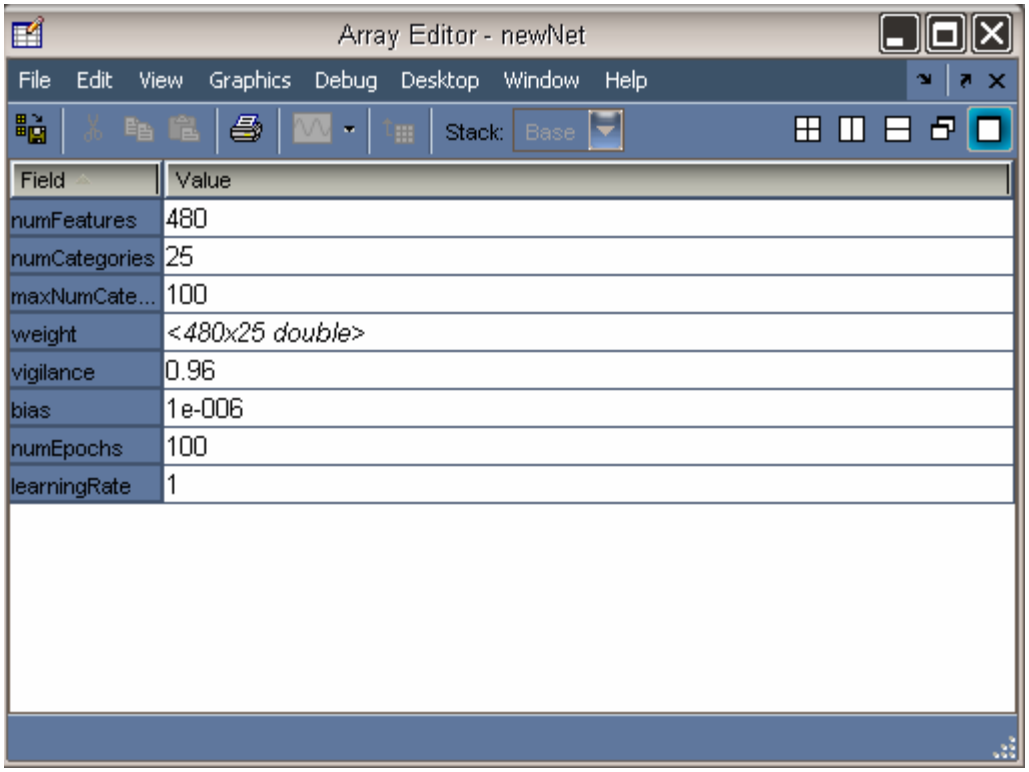


Figura 73 - Estructura de la red entrenada con ArtLearn

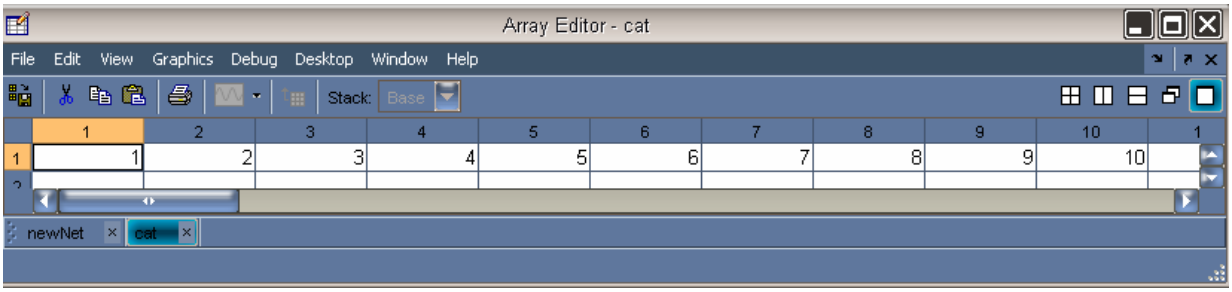


Figura 74 - Categorías creadas en el entrenamiento de la red.

Las columnas de 1 a 26 de la figura, representan a las letras de la A a la Z, en este ejemplo se crearon 25 categorías que se muestran en la única fila del vector.

El reconocimiento es muy similar al entrenamiento e la red, se captura la foto de la mano que se desea reconocer y se guarda en una variable, después se le aplica el procesamiento con la función proc_foto para obtener los tres erfiles de la imagen. Estos

perfiles son almacenados en forma de un vector, al cual se le aplica el complemento de datos para obtener la entrada a la red.
Con el vector de entrada se utiliza la función:

newCat = ART_Categorize(newNet, ccNewInput)

donde newNet es la estructura de la red entrenada y ccNewInput es el vector de entrada complementado.
El resultado queda guardado en la variable newCat que muestra la categoría a la cual pertenece la imagen, si ésta no es reconocida para ninguna categoría y ya no se puede crear una nueva porque se ha llegado al límite de máximo número de categorías, entonces presenta un valor de -1 que indica que no es reconocida.

Como ejemplo de entrenamiento y reconocimiento se utilizará una carpeta que contiene las imágenes de manos que representan las letras: a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z. Se tienen 26 letras en total, entonces se quiere obtener 26 categorías, una por cada letra.

El número de iteraciones realizadas ha sido de 109.

La red creada para estas imágenes es la siguiente:

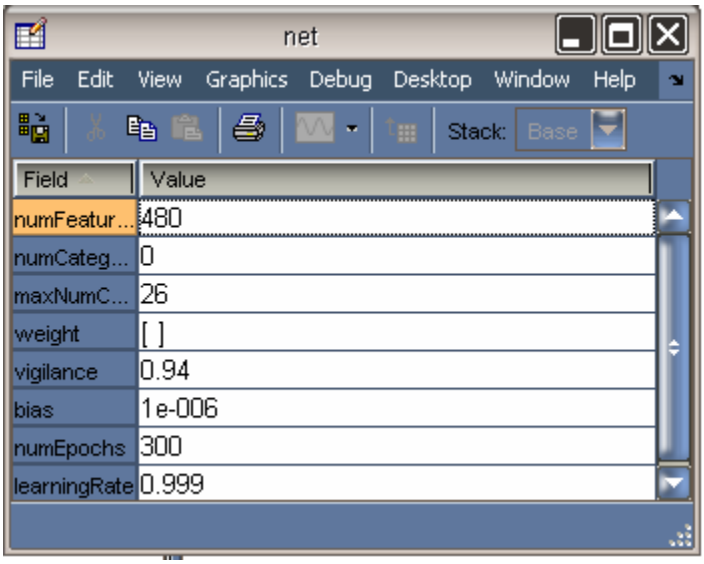


Figura 75 - Estructura de la red a entrenar

Las categorías creadas con los parámetros anteriores se muestran en la figura:

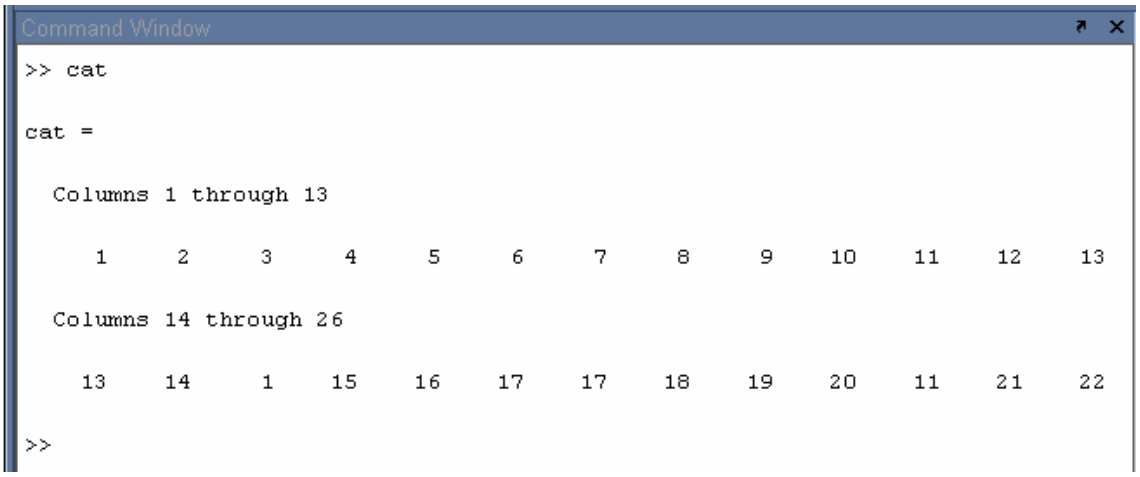


Figura 76 – Resultado de la creación de categorías

Se observa que las letras m y n fueron asignadas a la categoría 13, la letra a y p a la categoría 1, las letras s y t a la categoría 17 y las letras k y x a la categoría 11.

Si se trata de reconocer con letras distintas a las utilizadas en el entrenamiento se obtiene lo siguiente:

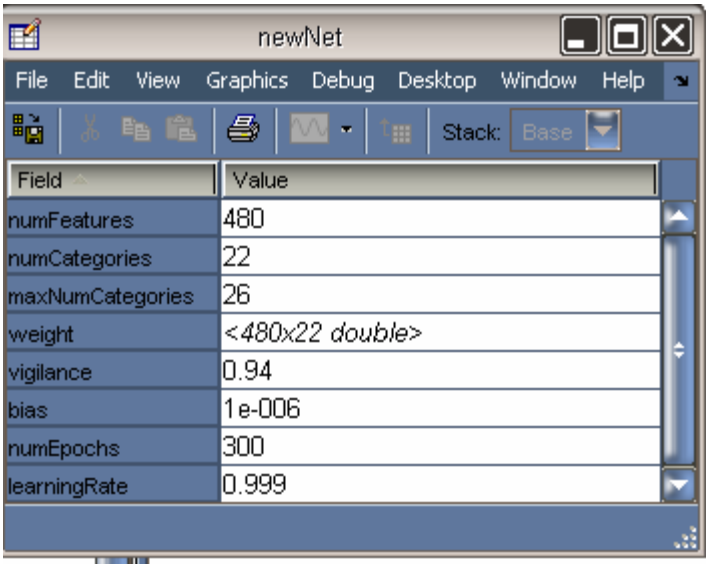


Figura 77 –Estructura de la red entrenada.

Se hará la prueba con las siguientes letras:

A B L

V M H

Figura 78 – Imágenes usadas para probar el resultado del entrenamiento de la red ART2

Los resultados de newCat para las imágenes anteriores son:

A: -1, indica que no reconoce la letra.

B: 2, la letra es asignada a la categoría 2 que es la correcta.

L: 12, La letra es asignada a categoría 12, fue reconocida.

V: -1, no reconoce.

M: 17, letra asignada a categoría 17, en este caso puede ser S o T, no reconoce.

H: 7, la letra fue asignada a categoría 7, la reconoció como la letra G.

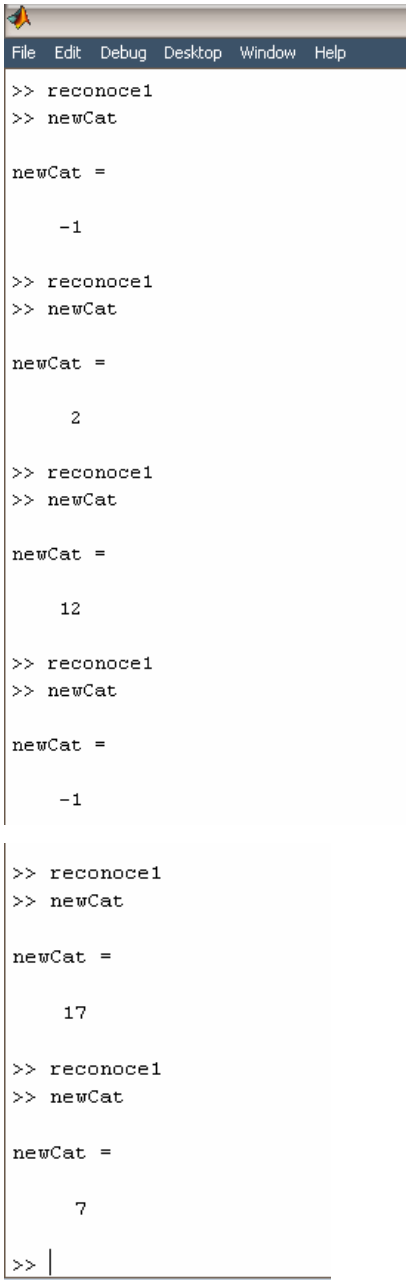


Figura 79 – Pruebas de reconocimiento

De 6 letras reconoció correctamente solo 2 por lo tanto hubo una efectividad de 33.3%. Para obtener resultados diferentes hay que variar el parámetro de vigilancia y la razón de aprendizaje y realizar nuevamente el reconocimiento.

VALORES PROPIOS (EIGEN-VALUES)

¿Qué son los Valores Propios o Auto-Valores (Eigen-Values)?

Si se realiza una acción que afecte la forma o la orientación de un objeto, por ejemplo estirar una banda elástica. Si se investiga a fondo matemáticamente las transformaciones que sufre el objeto, puede encontrarse que hay direcciones dentro de este que no cambian aún después de ocurrida la transformación. Retomando el ejemplo de la banda elástica, si se hubiera dibujado en ella una flecha, al desaparecer la fuerza que deforma la banda, ¿qué pasaría con la flecha?

La respuesta a la pregunta anterior depende de la forma de la flecha original.

Para ilustrar el ejemplo se muestra la Figura 80.

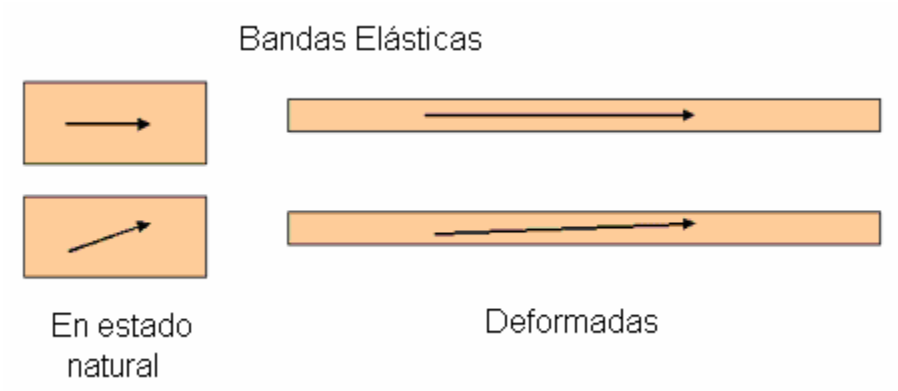


Figura 80 – Ejemplo de bandas elásticas para explicar el concepto de "Valores Propios"

A pesar de la transformación, la flecha conserva la misma dirección para unos casos y puede cambiar para otros. La dirección que no cambia se llama VECTOR PROPIO de la transformación y la cantidad asociada al estiramiento es un VALOR PROPIO.

Los valores propios son multiplicadores, son los números que representan cuánto se estiró la banda, es decir una medida del cambio de escala.

Para que tenga sentido hablar de valor propio, debe existir una “operación asociada”, es decir de la transformación que ha ocurrido y a la vez una dirección asociada (el vector propio).

Por ejemplo si compara los tamaños de dos reglas, y una de ellas es 2 veces más grande que la otra, entonces no basta con asumir que 2 es el valor propio por si mismo, es necesario definir que la operación asociada es “agrandar” y la dirección es “mayor”

Volviendo al ejemplo de la banda elástica, ésta tiene un vector propio en la dirección izquierda-derecha, porque esa flecha todavía señala en esa dirección aún después de la transformación (estiramiento) y el valor propio es la cantidad de veces que se incrementó la longitud de la banda con el estiramiento.

Aplicación de los Valores Propios en el reconocimiento de imágenes.

Debido a que los valores propios y vectores propios son obtenidos considerando variaciones en la forma, pueden ser utilizados en la identificación de imágenes que han sufrido cambios de intensidad, algún grado de rotación, cambio de escala u otro tipo de deformación, siempre dentro de rangos preestablecidos de tolerancia. Debe crearse una base de datos en la cual se encuentren cada una de las imágenes que se desea reconocer, además dentro de la misma base de datos se debe incluir variaciones de esas imágenes identificándolas como las originales, para que se encuentre la relación entre ellas y la deformación o cambio que han sufrido. A la base de datos con las imágenes (cada imagen en un vector) se la llamará en adelante “*base de manos*”.

Los datos que serán utilizados para la creación de la *base de manos* pueden ser obtenidos de dos tipos de imagen:

- Imagen Esqueletizada (binarizada con realce de bordes).
- Imagen en Escalas de Gris.

Para ambos casos, debe realizarse previamente el pre-procesamiento de la foto, para obtener solamente la información más importante (la mano recortada).

Además de encontrar los valores propios para cada categoría, se encuentra un vector llamado “vector al espacio de manos” que representa un vector de datos común a todas las imágenes contenidas en la base de manos, éste vector al espacio de manos sirve para identificar nuevas imágenes o aquellas que han sufrido cambios muy superiores a los rangos permitidos

3.3 Descripción del Funcionamiento.

3.3.1 Descripción del Procesamiento de la Imagen.

La tarea de reconocimiento de las señas comienza por realizar un procesamiento a la imagen capturada, con el fin de reducir las variaciones producto de cambios en el entorno. Ejemplos de estas variaciones son:

- Mayor o menor intensidad de luz en la habitación donde se capturan las imágenes;
- La distancia a la cual el usuario coloque la mano para realizar las señas;
- Cambio de webcam;
- Cambio de usuario.

Para que la descripción del proceso sea fácil de entender, se presentan imágenes que muestran el efecto de los procedimientos aplicados. La explicación detallada sobre las funciones y procesos que se llevan a cabo se presenta en forma de comentarios en los archivos de función de Matlab.

El procedimiento comienza capturando la imagen desde la webcam. Esto se logra con el uso de la caja de herramientas de adquisición de imágenes que contiene Matlab.

Para el procesamiento se necesita tener la imagen capturada en formato ‘rgb’ o en escala de gris, además las dimensiones de altura para la seña de las letras ‘a’ y ‘b’ previamente encontradas.

Como ejemplo, se tomará una imagen tomada con un entorno iluminado y otra con un entorno más oscuro, tal como se muestra en la Figura 81. Sin embargo, debe recordarse que para el uso de la aplicación “ROLES” debe tratar de controlarse el entorno, procurando mantener el lugar donde se realice la captura de imágenes con las condiciones de luminosidad adecuadas.

	
(a) Imagen de ejemplo con iluminación adecuada	(b) Imagen de ejemplo con iluminación baja

Figura 81 – Imágenes de ejemplo para ser procesadas.

3.3.1.1 Paso 1: Ajuste de la imagen con ecualizado del histograma.

Este paso se realizará siempre y cuando la imagen de entrada sea demasiado oscura. El criterio para aplicar el ecualizado del histograma es encontrar el valor promedio de intensidades en la imagen de entrada. En una imagen en escalas de gris, los valores posibles de intensidades están comprendidos en el rango de 0 (color negro) a 255 (color blanco), por lo tanto el valor promedio de intensidad de toda la imagen debe ser un valor dentro del mismo rango. El umbral para aplicar el ecualizado del histograma es un valor promedio de 100, lo que indica que la mayor parte de la imagen tiene intensidades muy oscuras.

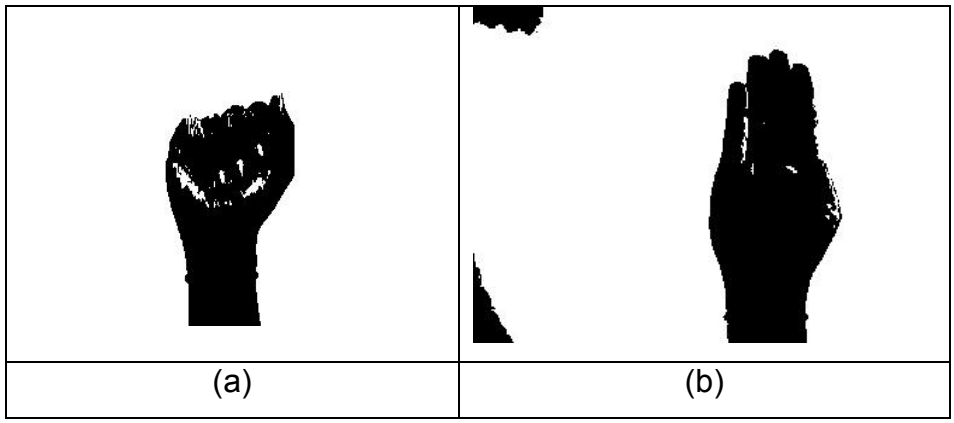
Para las imágenes de los ejemplo, no se aplica el ecualizado, ya que los promedios de intensidad son superiores al umbral (100).

Figura 70 (a) / intensidad promedio = 225;

Figura 8170 (b) / intensidad promedio = 160;

3.3.1.2 Paso 2: Binarización de la imagen con cálculo ajustable de umbral utilizado.

Este proceso consiste de cambiar la imagen de entrada a una en la cual toda el área ocupada por la mano y el brazo tengan un valor de “0” y el resto (lo que constituye el fondo de la imagen) tenga valor de “1”. Este procedimiento es aplicado para posteriormente hacer un recorte del área de interés (la usada por la mano).

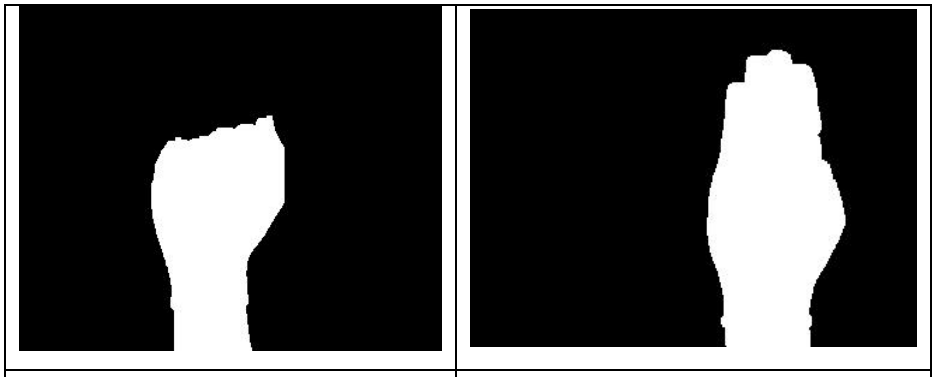


(a) (b)

Figura 82 –Imagen binarizada.

3.3.1.3 Paso 3: Mejoramiento de imagen binarizada.

De la imagen del paso 3 pueden hacerse varias observaciones, como por ejemplo que existen pequeñas áreas de la mano que presentan huecos. En otros casos pueden existir también pequeños puntos alrededor de la imagen (separados de la mano) como en forma de ruido. También la imagen puede presentar en las fronteras entradas del fondo que deben ser rellenadas por pertenecer al área de la mano. Para eliminar los problemas mencionados anteriormente se realizan operaciones de dilatación, erosión y abertura para mejorar la imagen, además se intercambian los valores del fondo (toma valores de “0”) y la mano (toma valores de “1”).

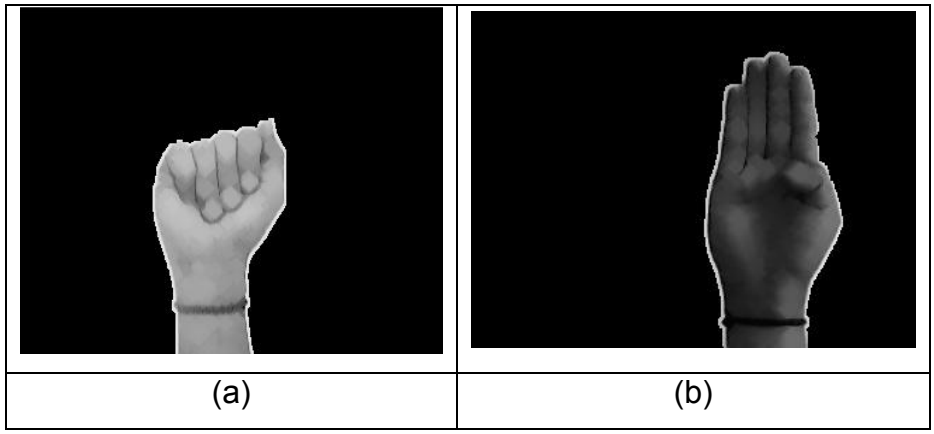


(a) (b)

Figura 83 – Definición del área que ocupa la mano.

3.3.1.4 Paso 4: Eliminación de objetos en el fondo en la imagen en escala de gris.

Haciendo uso de la imagen obtenida en el paso 3, se eliminan los objetos que no pertenezcan a la mano, realizando una multiplicación píxel por píxel entre la Figura 81 y la Figura 83. Previamente a la operación de multiplicación, se realiza un "suavizado" de la imagen es escala de gris mediante un filtrado, con el objetivo de disminuir los cambios bruscos de intensidad.



(a) (b)

Figura 84 – Imagen en escala de grises con fondo eliminado.

3.3.1.5 Paso 5: Encontrar las coordenadas del centroide de la mano.

A partir de la imagen de la Figura 83 se encuentran las coordenadas del centroide de la mano y con estas coordenadas se encuentran los extremos o fronteras de la mano, tanto para la izquierda como para la derecha.

En este paso, debe mencionarse que como producto de una imagen de entrada con poca iluminación pueden generarse en el proceso de binarización otros objetos en las esquinas de la imagen. El inconveniente que provocan estos objetos es una mayor cantidad de centroides (uno para cada objeto), por lo que se hace necesario identificar el centroide que se ubique lo más cerca posible del centro de la imagen de entrada.

Debe recordarse también que dentro de las consideraciones del uso de la aplicación "ROLES" se manifiesta que el usuario debe tratar de que la mano cubra la mayor área de la foto capturada, así como que su ubicación (de la mano) dentro de la foto sea lo más centrada posible.

Figura 8372 – (a) / Centroide = [164 , 185];
Figura 8372 – (b) / Centroide = [216 , 139];

3.3.1.6 Paso 6: Recorte de áreas laterales vacías.

Conociendo las coordenadas del centroide, se procede a encontrar los límites de la imagen hacia los lados (eje horizontal). Esto se logra encontrando un vector fila que contiene un valor de "1" para las columnas que contienen información (área de la mano). De este vector se encuentran las posiciones más próximas al centroide donde se da cambio de "1" a "0", de la siguiente forma:

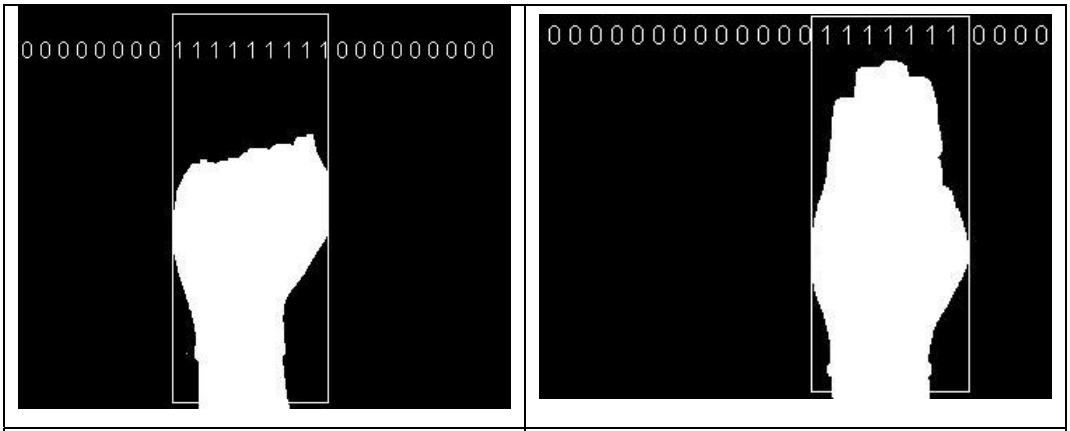


Figura 85 – Definición de zona usada por la mano indicada por valor "1".

Con la identificación de las coordenadas de los límites de la mano, se procede a realizar el corte, tomando de la matriz que define la imagen las columnas comprendidas entre el límite izquierdo y el límite derecho, teniendo como resultado lo mostrado en la Figura 86.

3.3.1.7 Paso 7: Esqueletizado de la mano.

Se encuentra la imagen de la mano esqueletizada con un extractor de bordes (canny), a partir de la imagen en escala de gris, con lo que se obtiene lo mostrado en la Figura 87. El proceso de esqueletizado consiste básicamente de realizar una identificación de las líneas que describen en la imagen un cambio de intensidad considerablemente notable para el ojo humano.

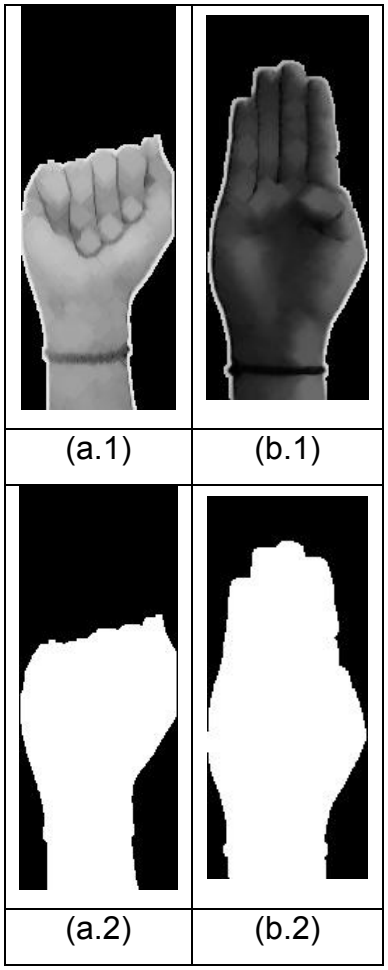


Figura 86 – Imágenes de mano con lados vacíos eliminados:

(a.1) y (b.1) corresponden a las imágenes de escala de gris con los lados eliminados;
(a.2) y (b.2) corresponden a las imágenes binarizadas con los lados eliminados.

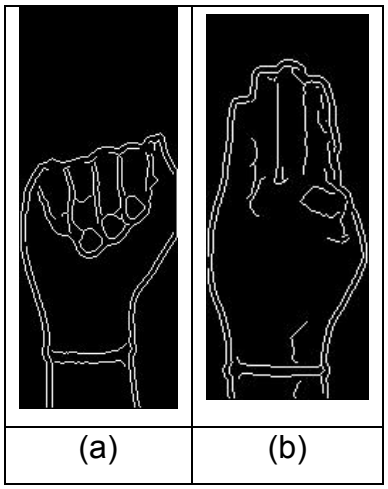


Figura 87 – Imagen esqueletizada a partir de la imagen en escala de grises.

3.3.1.8 Paso 8: Ubicación del borde superior.

Se encuentra el valor con el cual se realiza el recorte del área de arriba que no es parte de la mano. El procedimiento utilizado para encontrar el límite superior es el mismo descrito para los límites laterales con la variante de crear un vector fila (no columna como en el paso 6), con valor de “1” para todas las columnas que contienen información del área de la mano.

El recorte se aplica a las 3 imágenes: en escala de gris, la binarizada y la esqueletizada.

3.3.1.9 Paso 9: Ubicación del límite inferior.

Como último recorte de la imagen se tiene la eliminación de la parte inferior. Para hacer más fácil esta tarea se adoptó la convención en el uso de la aplicación “ROLES” de usar siempre para el reconocimiento de las señas una cinta elástica de color negro (u otro color oscuro) en la muñeca. Con la cinta mencionada se generan líneas en la imagen esqueletizada que pueden ser filtradas con el fin de encontrar la posición donde se encuentra y así cortar la mano por la parte inferior hasta esas coordenadas.
A la imagen esqueletizada se le aplica un filtro para resaltar las líneas horizontales, luego se eliminan los puntos más pequeños de la figura resultante y se encuentra la posición de la línea horizontal en la parte de abajo.

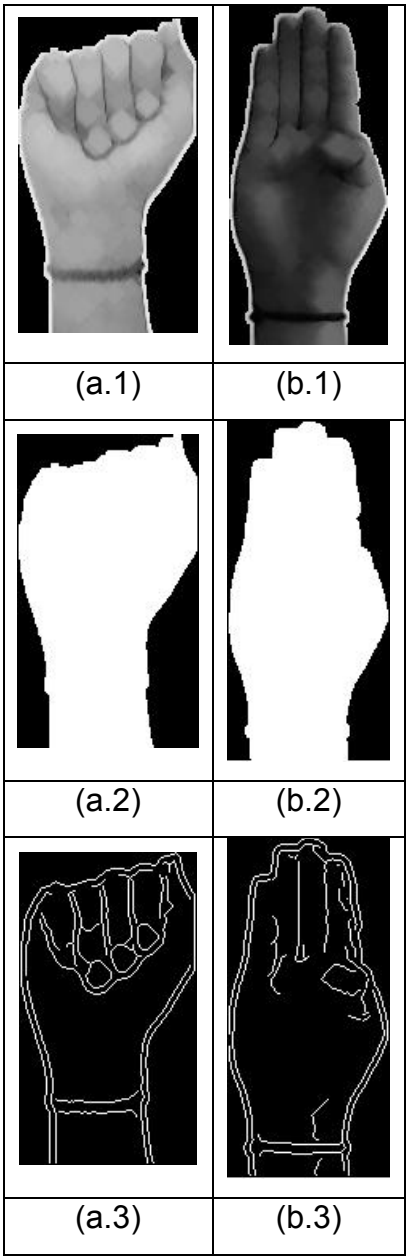
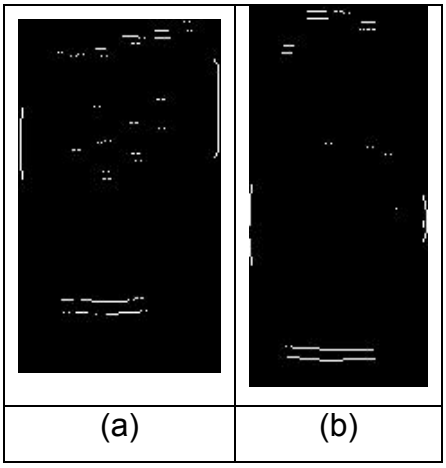


Figura 88 – Resultado del recorte de la parte superior para la imagen en intensidades de gris, la esqueletizada y la de dos zonas (mano y fondo).



(a) (b)

Figura 89 – Imagen esqueletizada con realce de líneas horizontales para identificar límite inferior.

3.3.1.10 Paso 10: Medición de la muñeca.

Se toma la región inferior de la imagen esqueletizada y con ella se encuentra el ancho de la muñeca para establecer una referencia o escala, respecto a la cual puedan ser medidas otras dimensiones de la mano, como el ancho y la altura. La forma de encontrar el ancho es aplicar un filtro para resaltar las líneas verticales del segmento de la muñeca, con ello se encuentran las posiciones de las dos líneas más lejanas y se calcula la diferencia de posición en coordenadas horizontales (distancia).

(a) (b)

Figura 90 – Franja tomada de la imagen esqueletizada con realce de líneas para realizar la medición del ancho de la muñeca.

3.3.1.11 Paso 11: Agregando área vacía arriba de la mano.

Una vez encontrado el ancho de la muñeca según el paso 10, se calcula la altura de la mano, esta distancia es comparada con la altura de referencia para averiguar si se encuentra dentro de los márgenes aceptables, al determinar que cumple con las condiciones aceptables se procede a recortar la mano por abajo, tal como se indica en la Figura 91.

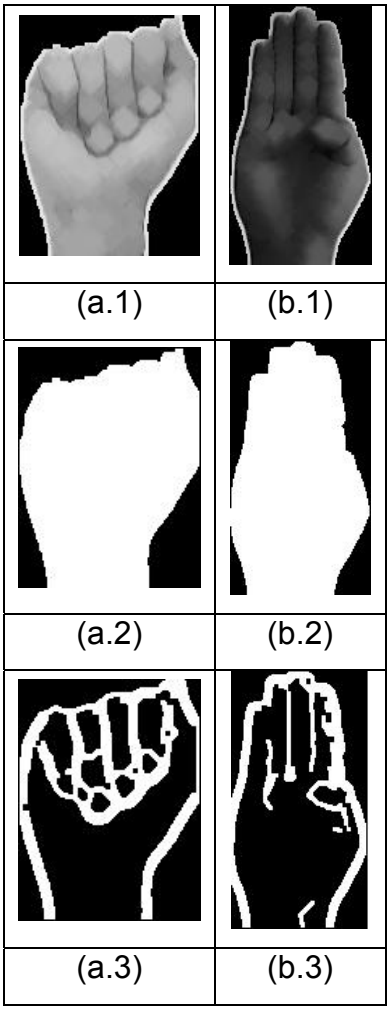


Figura 91 – Imágenes resultantes después de eliminar la parte inferior por debajo de la marca (usando el límite definido por la cinta elástica negra en la muñeca).

De acuerdo a las dimensiones (alto y ancho) de la mano, encontradas tomando el ancho de la muñeca como referencia, se estima si debe agregarse a la imagen una zona vacía en la parte de arriba, como en el caso de las señas que se representan con el puño.

La cantidad de filas de fondo que deben agregarse es calculada considerando dos dimensiones básicas:

- La altura máxima posible, obtenida de la foto con la seña correspondiente a la letra “b” ya que en esta se deben colocar totalmente extendidos los dedos.
- La altura mínima posible, obtenida de la foto con la seña correspondiente a la letra “a” porque en esta se encuentran la mano totalmente cerrada (en forma de puño).

La cantidad de filas se calcula con el siguiente criterio:

Cantidad de filas agregadas = Altura máxima de mano – Altura actual de mano

Altura actual de mano = límite superior – límite inferior

Todo calculado utilizando la escala del ancho de la mano, para mantener la relación de aspecto, aunque la foto sea más pequeña.

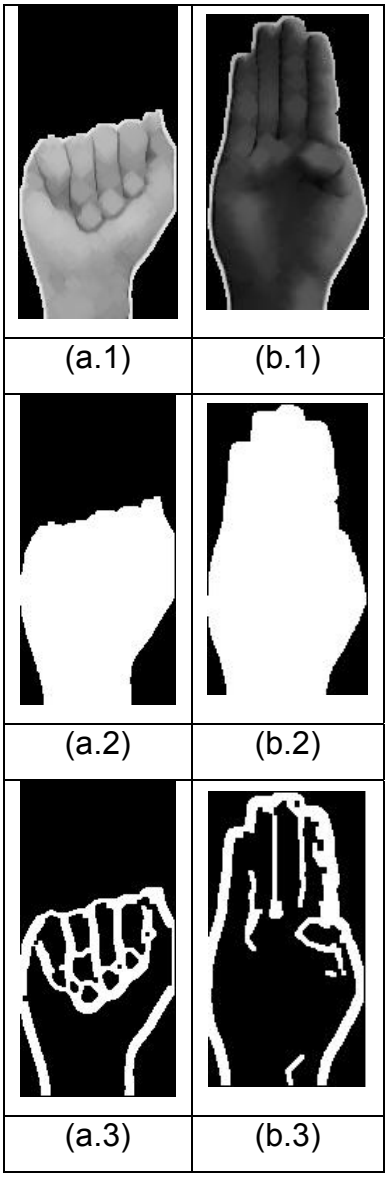


Figura 92 – Imágenes resultantes después de agregar el área vacía superior (para las fotos de manos cerradas).

3.3.1.12 Paso 12: Encontrando los Perfiles de la mano.

A partir de la imagen binarizada (Figura 92 – (a.2)) que contiene el área usada por la mano (color blanco) y el fondo (color negro) se encuentran los vectores de distancias desde la izquierda, desde arriba y desde la derecha hasta la mano.

Ya que la imagen puede tener tamaños diferentes producto de cambios en la resolución de la webcam, o que el usuario aumentó o disminuyó la distancia entre la webcam y su mano, provocando un cambio de escala de la foto de la mano. Para que estos vectores tengan siempre la misma longitud se realiza un redimensionado de la imagen a 90 filas y 60 columnas para que además se conserve una relación de aspecto promedio. De esta forma se obtienen 3 vectores correspondientes a los 3 perfiles descritos como:
Perfil Izquierdo (p_I),
Perfil Arriba (p_A),
Perfil Derecho (p_D).

Cada uno de los perfiles se normaliza para un valor máximo de 1.

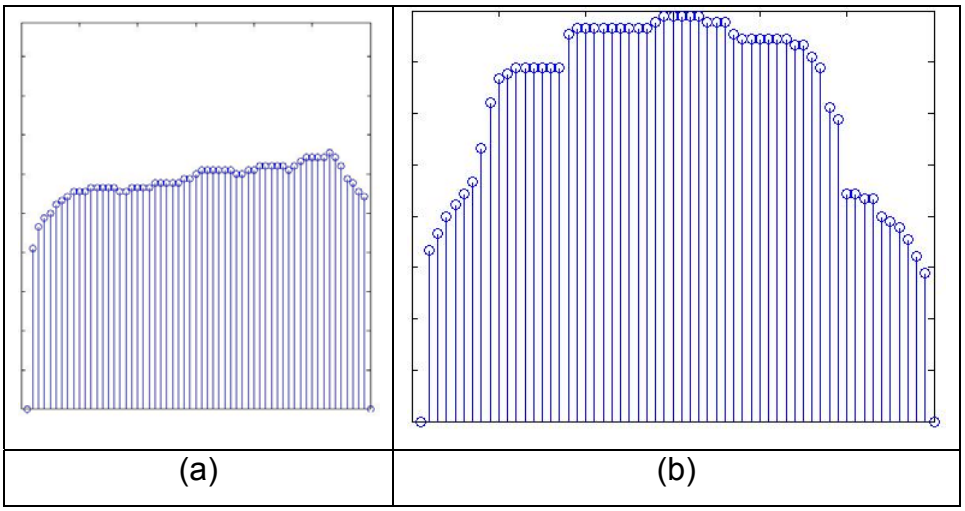


Figura 93 – Gráfico que representa el vector de distancias (perfil Arriba) obtenidas desde el eje de arriba hasta la mano.

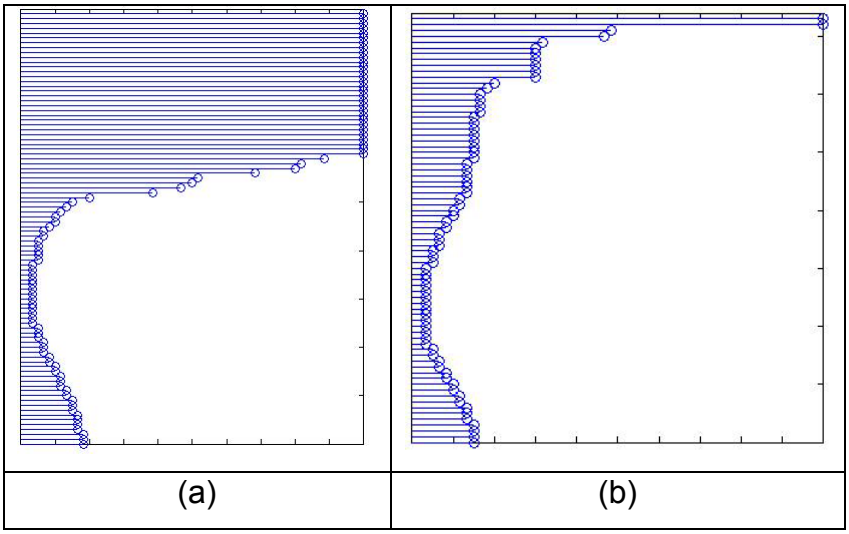


Figura 94 - Gráfico que representa el vector de distancias (perfil Izquierdo) obtenidas desde el eje de la izquierda hasta la mano.

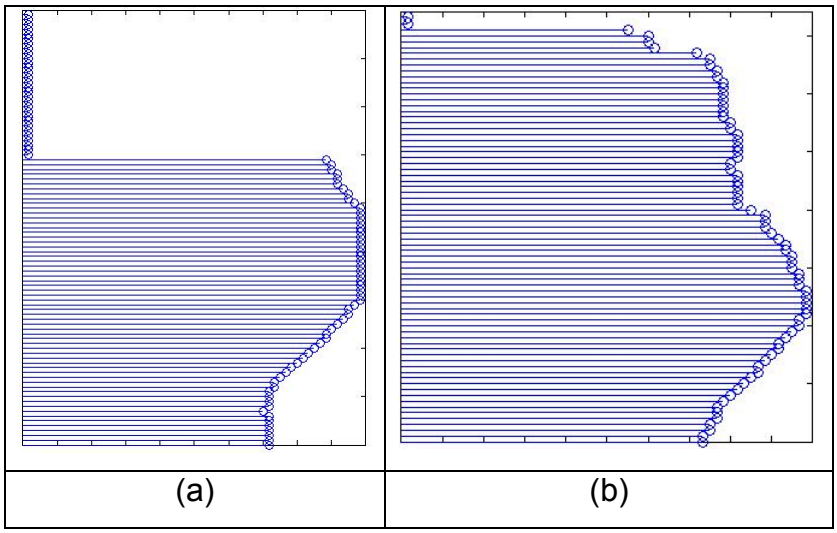


Figura 95 - Gráfico que representa el vector de distancias (perfil Derecho) obtenidas desde el eje de la derecha hasta la mano.

La función (archivo 'm') que realiza todos estos procesos tiene el nombre de: 'proc_foto.m', dentro de ella se encuentran comentarios para explicar detalladamente todos los procedimientos descritos en cada paso.

3.4 PRUEBAS DE EFECTIVIDAD DEL PROGRAMA ROLES

3.4.1 MÉTODO COMBINADO.

Se realizaron las pruebas con el método combinado de redes neuronales y valores propios de las imágenes. Se utilizarán 10 carpetas de fotos, cada una contiene 26 imágenes, no se incluirá la letra ñ. Las carpetas 1 a la 4 contienen fotos de cuatro personas distintas en donde el fondo es más oscuro y algunas letras han sido rotadas de la posición con la que fueron entrenadas las redes. Las otras 6 carpetas contienen las fotos que tienen un fondo más claro, y la posición de las manos formando las letras es igual a la que tenían las fotos de manos con que fue entrenada la red. El valor de ecualizado será de -0.04 para todas las pruebas.

LETRA	Carpeta 1	Carpeta 2	Carpeta 3	Carpeta 4	Carpeta 5
A	No reconoce	Reconoce	No reconoce	Reconoce	Reconoce
B	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
C	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
D	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
E	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
G	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
H	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
I	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
J	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
K	Reconoce	No reconoce	Reconoce	No reconoce	Reconoce
L	Reconoce	No reconoce	No reconoce	No reconoce	Reconoce
M	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	Reconoce	Reconoce	Reconoce
O	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce

R	Reconoce	No reconoce	No reconoce	No reconoce	Reconoce
S	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
T	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
U	No reconoce	No reconoce	No reconoce	Reconoce	Reconoce
V	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
W	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
X	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
Y	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
Z	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce

LETRA	**Carpeta 6**	**Carpeta 7**	**Carpeta 8**	**Carpeta 9**	**Carpeta 10**
A	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
B	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
C	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
D	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
E	No reconoce	Reconoce	Reconoce	Reconoce	No reconoce
F	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
G	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
H	No reconoce	No reconoce	No reconoce	Reconoce	No reconoce
I	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
J	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
K	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
L	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
M	Reconoce	No reconoce	Reconoce	No reconoce	Reconoce
N	Reconoce	No reconoce	Reconoce	Reconoce	Reconoce
O	No reconoce	Reconoce	No reconoce	Reconoce	No reconoce
P	No reconoce	No reconoce	Reconoce	Reconoce	Reconoce
Q	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
R	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce

S	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
T	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
U	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
V	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
W	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
X	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
Y	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
Z	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce

Total de reconocidas:

Carpeta 1: 5
Carpeta 2: 5
Carpeta 3: 8
Carpeta 4: 4
Carpeta 5: 23
Carpeta 6: 21
Carpeta 7: 21
Carpeta 8: 22
Carpeta 9: 23
Carpeta 10: 19

TOTAL = 151

Efectividad fotos oscuras = (22/104)*100 = 21.15%
Efectividad fotos claras = (129/156)*100 = 82.69%

Las imágenes muestran los resultados del reconocimiento de las letras de cada carpeta, el campo de texto muestra los resultados obtenidos desde la A hasta la Z.

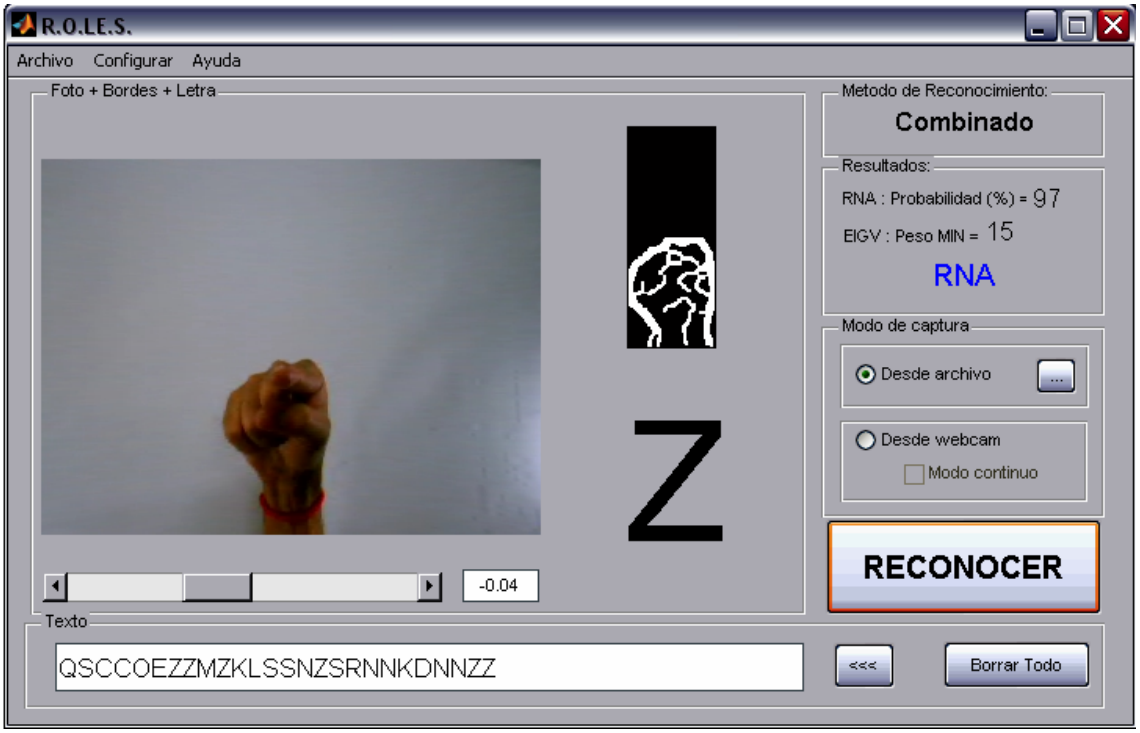


Figura 96 - Resultados de carpeta 1

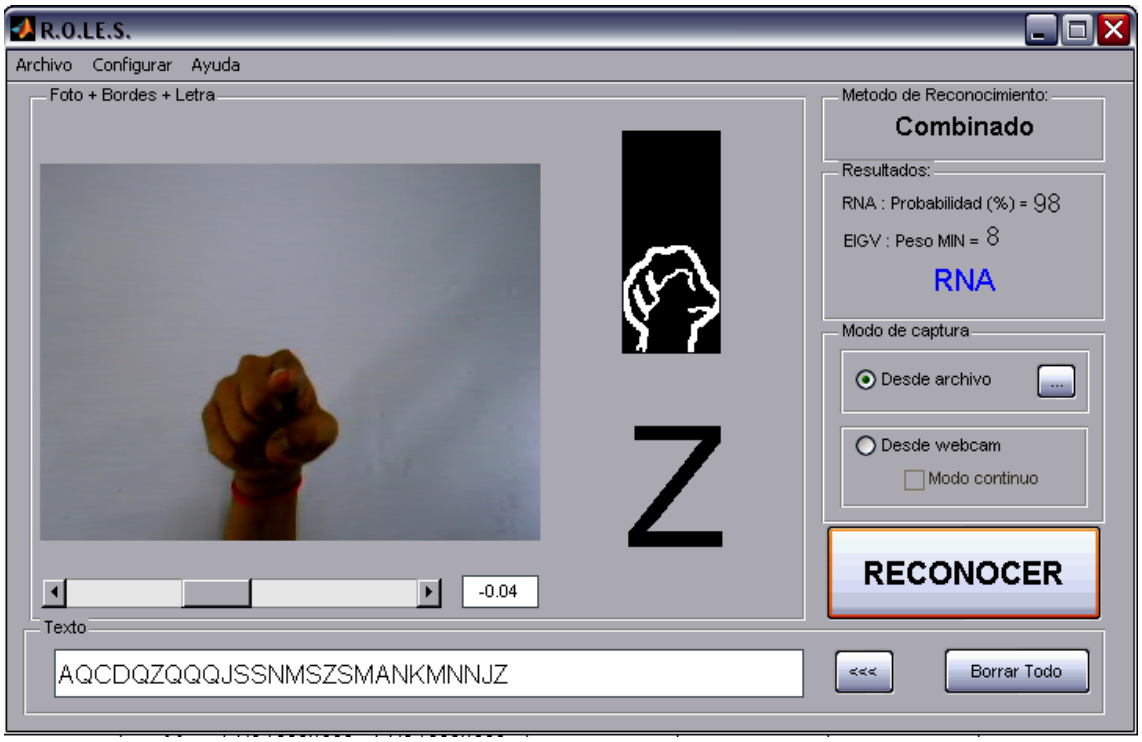


Figura 97 – Resultados de la carpeta 2

Figura 98 – Resultados de carpeta 3

Figura 99 – Resultados de carpeta 4

Figura 100 - Resultados de carpeta 5.

Figura 101 - Resultados de carpeta 6

Figura 102 - Resultados de carpeta 7

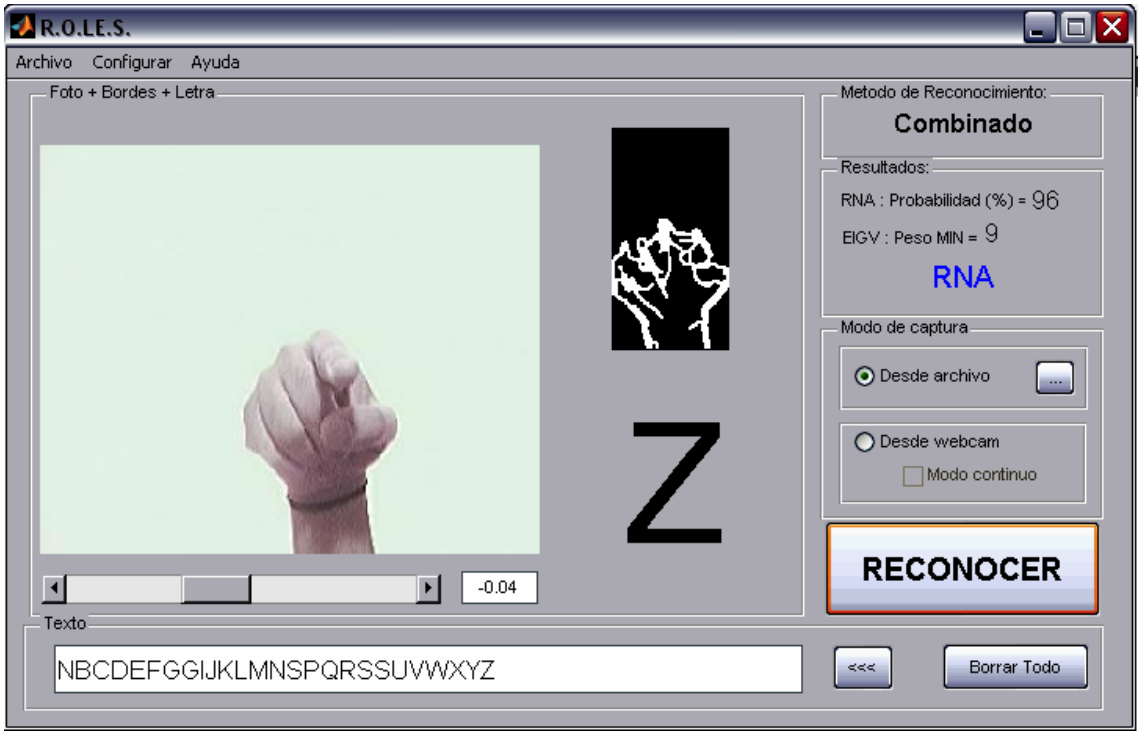


Figura 103 - Resultados de carpeta 8

Figura 104 - Resultados de carpeta 9

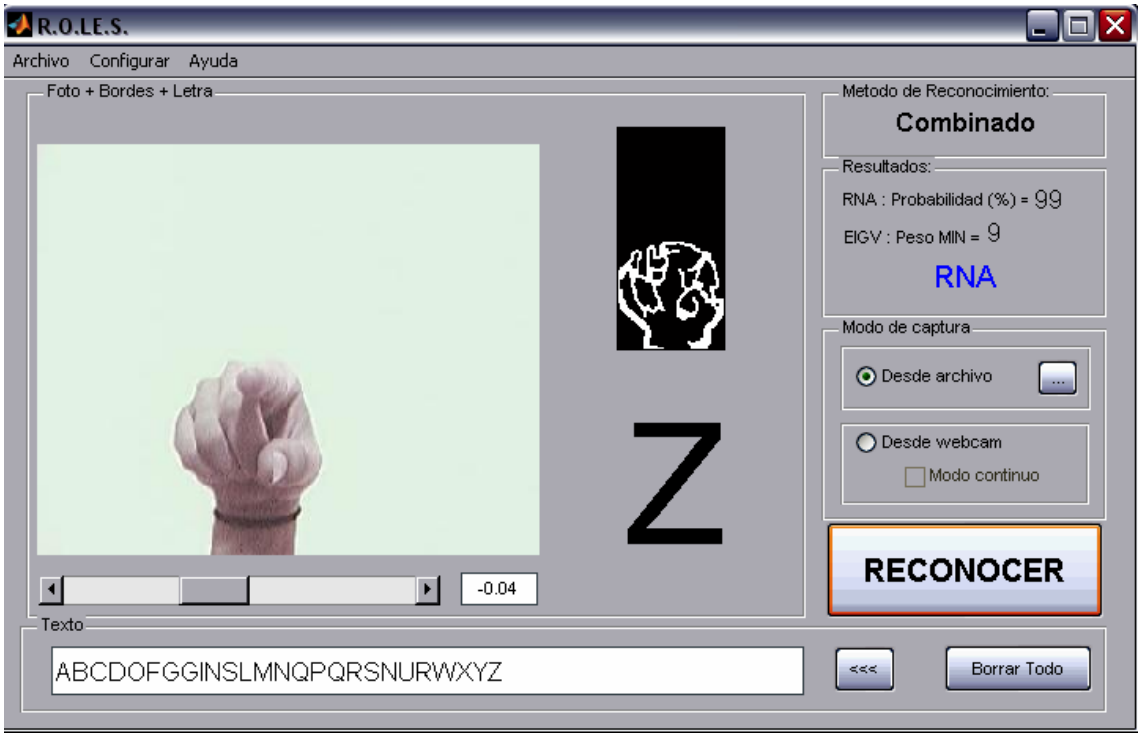


Figura 105 -Resultados de carpeta 10

3.4.2 MÉTODO RED NEURONAL

LETRA	Carpeta 1	Carpeta 2	Carpeta 3	Carpeta 4	Carpeta 5
A	No reconoce	Reconoce	No reconoce	Reconoce	Reconoce
B	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
C	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
D	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
E	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
F	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
G	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
H	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
I	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
J	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
K	Reconoce	No reconoce	Reconoce	No reconoce	Reconoce

L	Reconoce	No reconoce	No reconoce	No reconoce	Reconoce
M	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
N	No reconoce	No reconoce	Reconoce	Reconoce	Reconoce
O	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
Q	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
R	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
S	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
T	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
U	No reconoce	No reconoce	No reconoce	Reconoce	Reconoce
V	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
W	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
X	No reconoce	No reconoce	Reconoce	No reconoce	Reconoce
Y	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
Z	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce

LETRA	**Carpeta 6**	**Carpeta 7**	**Carpeta 8**	**Carpeta 9**	**Carpeta 10**
A	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
B	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
C	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
D	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
E	No reconoce	Reconoce	Reconoce	Reconoce	No reconoce
F	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
G	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
H	No reconoce	Reconoce	No reconoce	Reconoce	No reconoce
I	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
J	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
K	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
L	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
M	Reconoce	No reconoce	Reconoce	No reconoce	Reconoce

N	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
O	No reconoce	Reconoce	No reconoce	Reconoce	No reconoce
P	No reconoce	No reconoce	Reconoce	Reconoce	Reconoce
Q	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
R	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
S	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
T	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
U	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
V	Reconoce	Reconoce	Reconoce	Reconoce	No reconoce
W	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
X	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
Y	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce
Z	Reconoce	Reconoce	Reconoce	Reconoce	Reconoce

Total de reconocidas:

Carpeta 1: 5

Carpeta 2: 6

Carpeta 3: 8

Carpeta 4: 4

Carpeta 5: 23

Carpeta 6: 21

Carpeta 7: 24

Carpeta 8: 22

Carpeta 9: 23

Carpeta 10: 19

TOTAL = 155

Efectividad fotos oscuras = (23/104)*100 = 22.12%

Efectividad fotos claras = (132/156)*100 = 84.62%

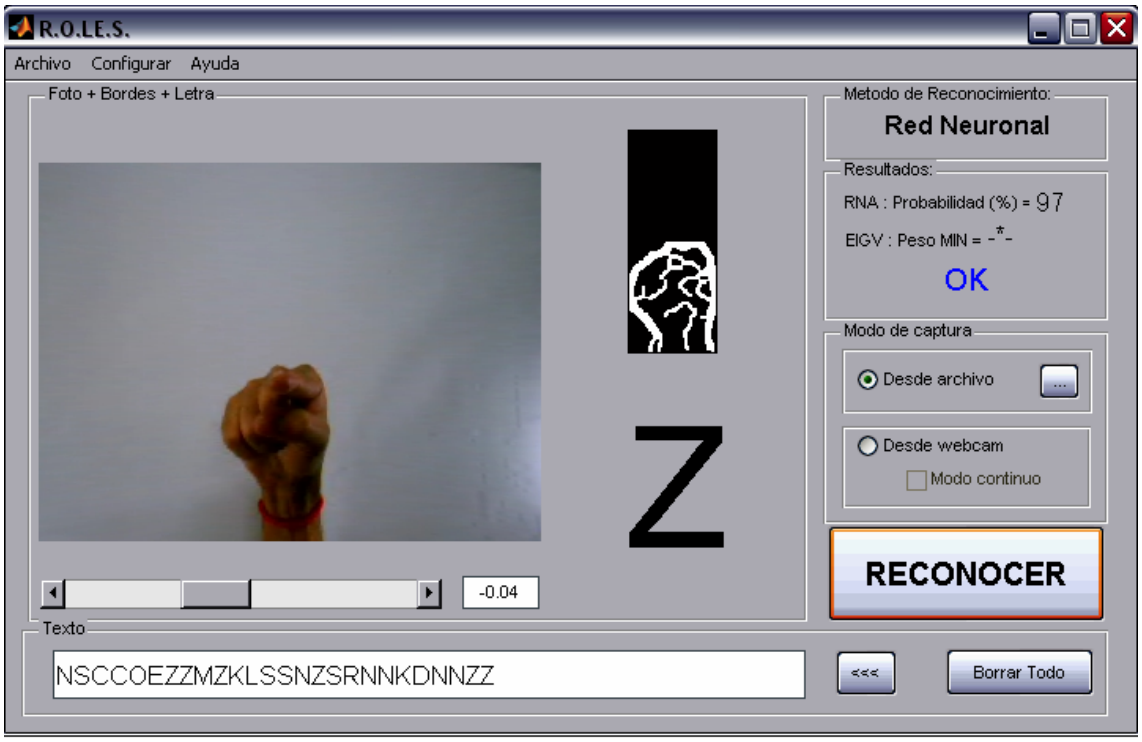


Figura 106 - Resultados de carpeta 1

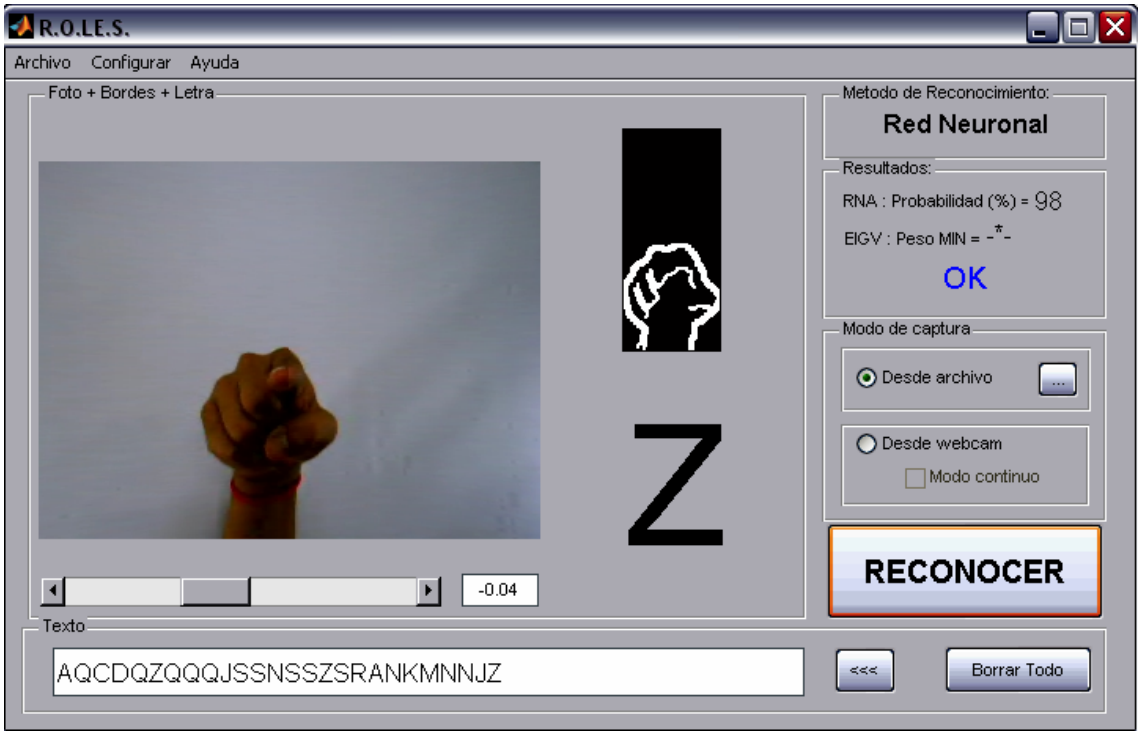


Figura 107 - Resultados de carpeta 2

Figura 108 - Resultados de carpeta 3

Figura 109 - Resultados de carpeta 4

Figura 110 - Resultados de carpeta 5

Figura 111 - Resultados de carpeta 6

Figura 112 - Resultados de carpeta 7

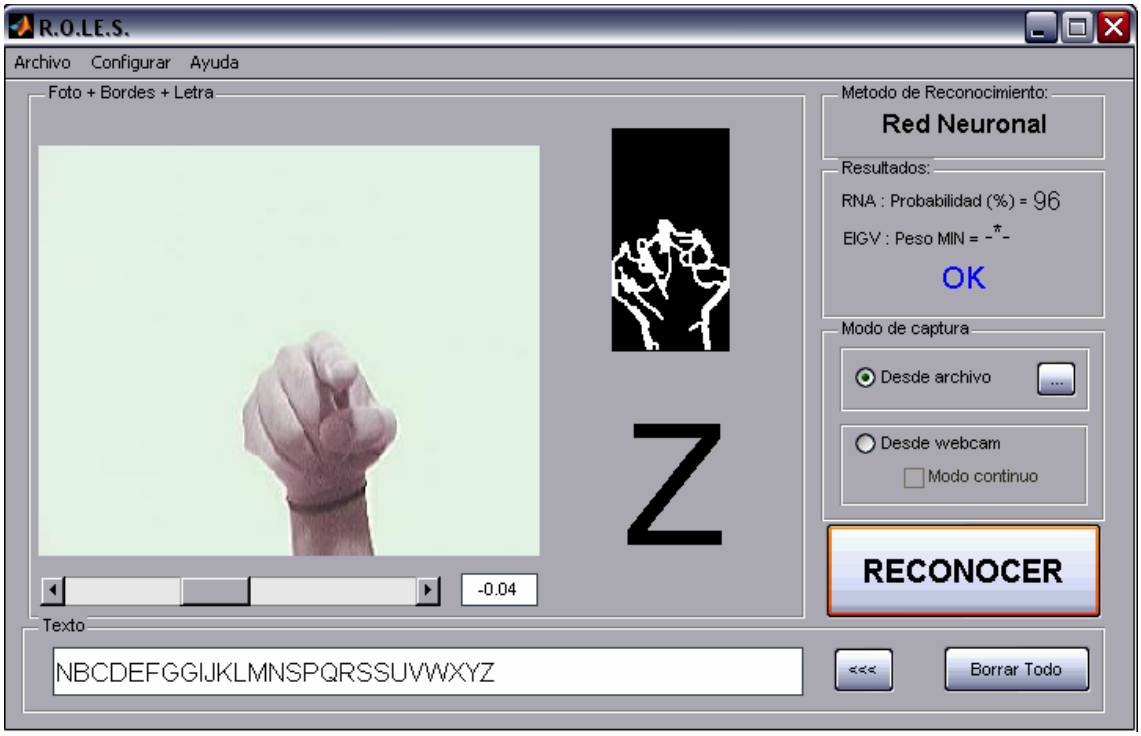


Figura 113 - Resultados de carpeta 8

Figura 114 - Resultados de carpeta 9

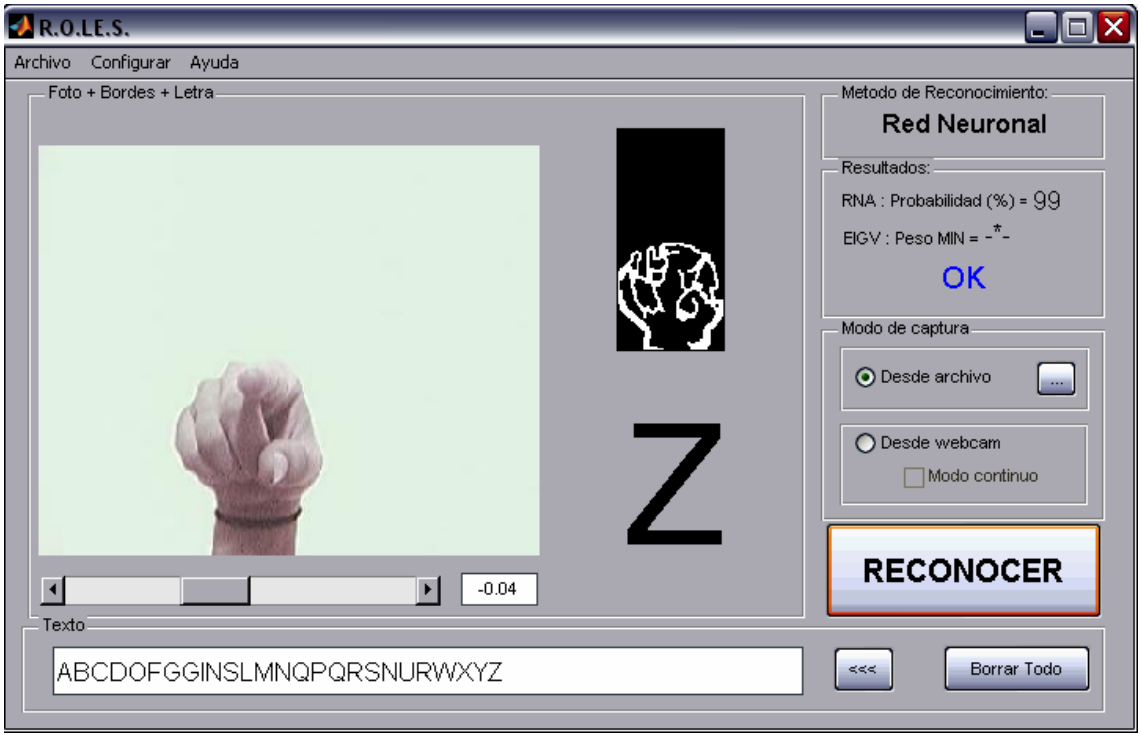


Figura 115 - Resultados de carpeta 10

3.4.3 MÉTODO DE VALORES PROPIOS

LETRA	Carpeta 1	Carpeta 2	Carpeta 3	Carpeta 4	Carpeta 5
A	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
B	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
C	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
D	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
E	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
F	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
G	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
H	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
I	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
J	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
K	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
L	Reconoce	No reconoce	No reconoce	No reconoce	Reconoce
M	No reconoce	No reconoce	Reconoce	Reconoce	No reconoce
N	No reconoce	No reconoce	Reconoce	No reconoce	No reconoce
O	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
P	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
Q	Reconoce	No reconoce	No reconoce	Reconoce	No reconoce
R	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
S	No reconoce	Reconoce	No reconoce	Reconoce	No reconoce
T	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
X	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
Y	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
Z	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce

LETRA	Carpeta 6	Carpeta 7	Carpeta 8	Carpeta 9	Carpeta 10
A	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
B	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
C	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
D	No reconoce	Reconoce	No reconoce	Reconoce	No reconoce
E	No reconoce	No reconoce	No reconoce	No reconoce	Reconoce
F	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
G	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
H	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
I	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
J	Reconoce	No reconoce	Reconoce	No reconoce	Reconoce
K	No reconoce	Reconoce	No reconoce	No reconoce	No reconoce
L	Reconoce	Reconoce	No reconoce	Reconoce	No reconoce
M	Reconoce	No reconoce	No reconoce	No reconoce	No reconoce
N	Reconoce	No reconoce	No reconoce	No reconoce	No reconoce
O	No reconoce	Reconoce	No reconoce	Reconoce	Reconoce
P	Reconoce	Reconoce	No reconoce	No reconoce	Reconoce
Q	No reconoce	No reconoce	No reconoce	Reconoce	No reconoce
R	Reconoce	Reconoce	No reconoce	No reconoce	No reconoce
S	No reconoce	Reconoce	No reconoce	No reconoce	Reconoce
T	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
U	No reconoce	No reconoce	No reconoce	No reconoce	No reconoce
V	No reconoce	No reconoce	Reconoce	Reconoce	No reconoce
W	No reconoce	No reconoce	No reconoce	Reconoce	No reconoce
X	No reconoce	No reconoce	Reconoce	Reconoce	No reconoce
Y	No reconoce	Reconoce	Reconoce	No reconoce	Reconoce
Z	No reconoce	No reconoce	Reconoce	No reconoce	No reconoce

Total de reconocidas:

Carpeta 1: 3

Carpeta 2: 4

Carpeta 3: 3

Carpeta 4: 3

Carpeta 5: 8

Carpeta 6: 6

Carpeta 7: 11

Carpeta 8: 5

Carpeta 9: 7

Carpeta 10: 6

TOTAL = 56

Efectividad fotos oscuras = (13/104)*100 = 12.50%

Efectividad fotos claras = (43/156)*100 = 27.56%

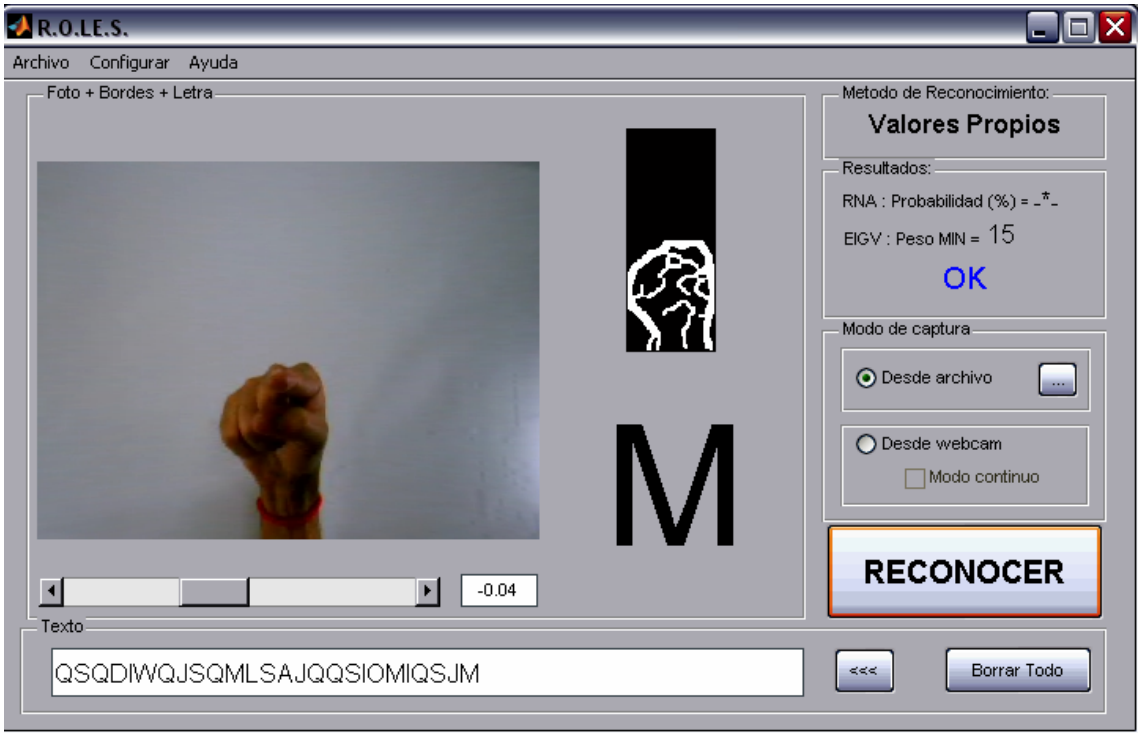


Figura 116 - Resultados de carpeta 1

Figura 117 - Resultados de carpeta 2

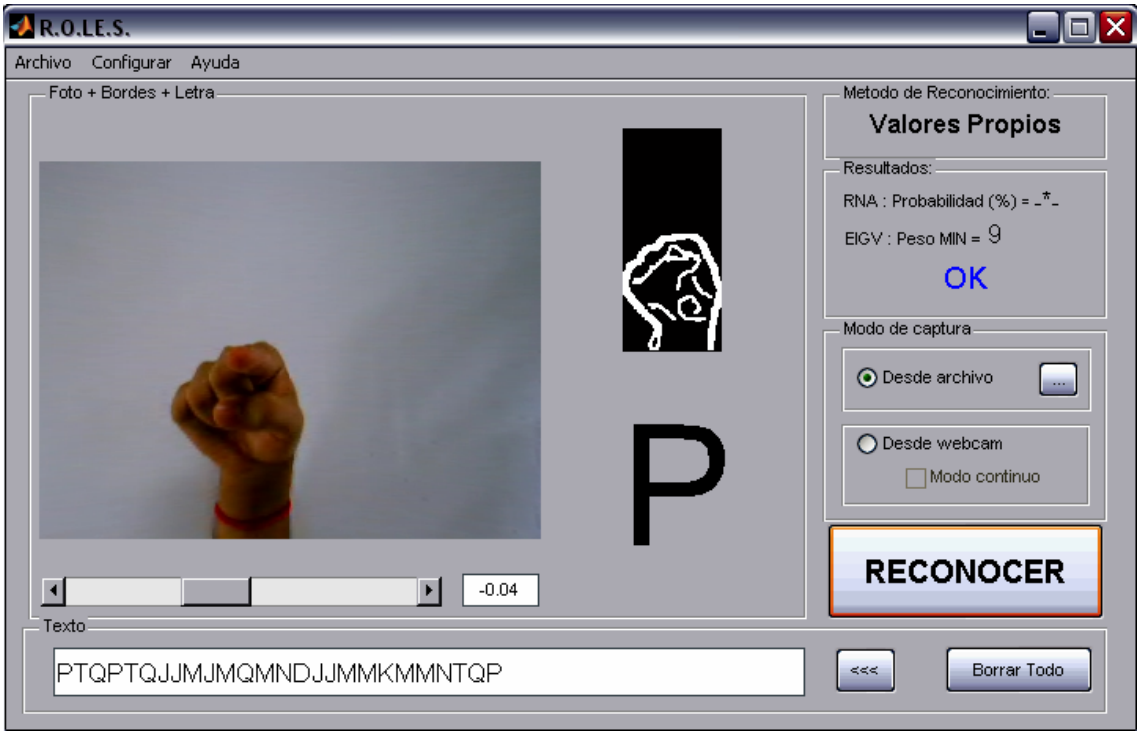


Figura 118 - Resultados de carpeta 3

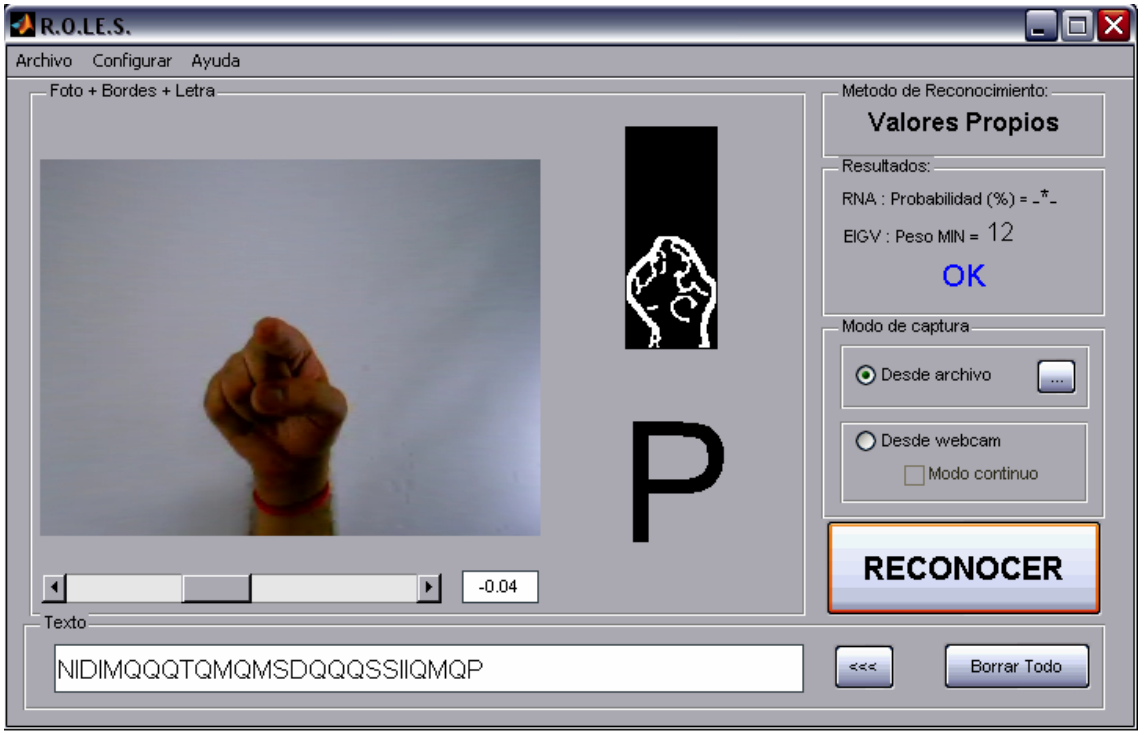


Figura 119 - Resultados de carpeta 4

Figura 120 - Resultado de carpeta 5

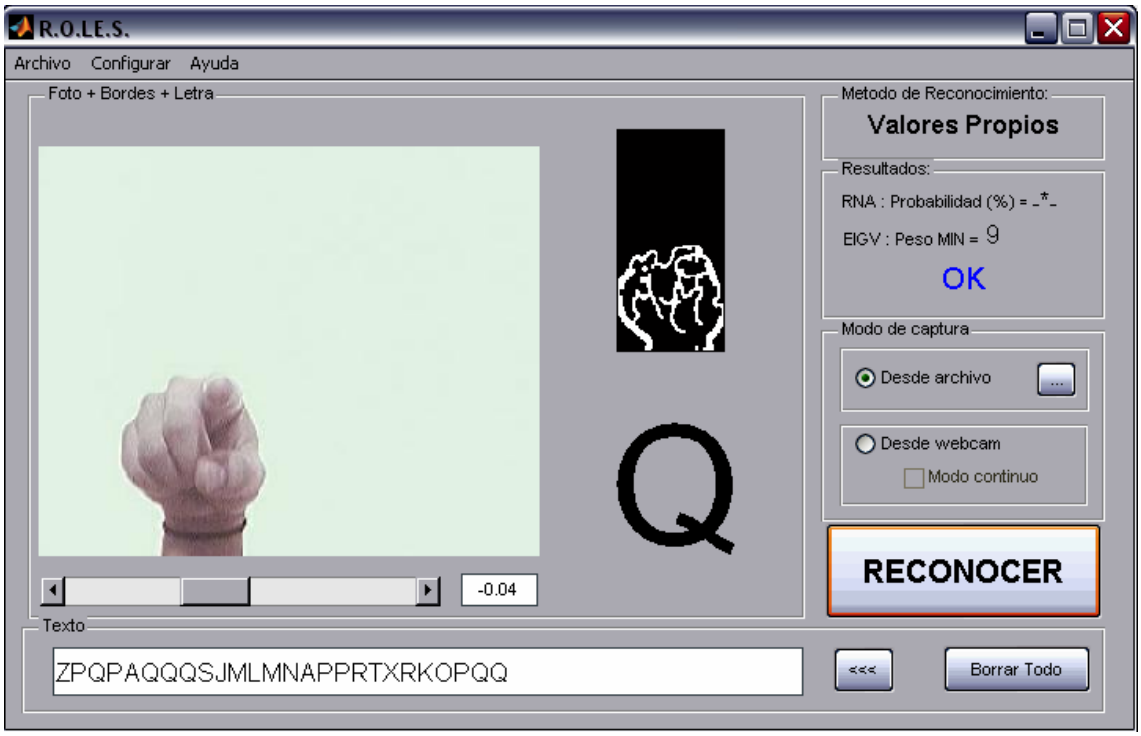


Figura 121 - Resultado de carpeta 6

Figura 122 - Resultado de carpeta 7

Figura 123 - Resultados de carpeta 8

Figura 124 - Resultado de carpeta 9

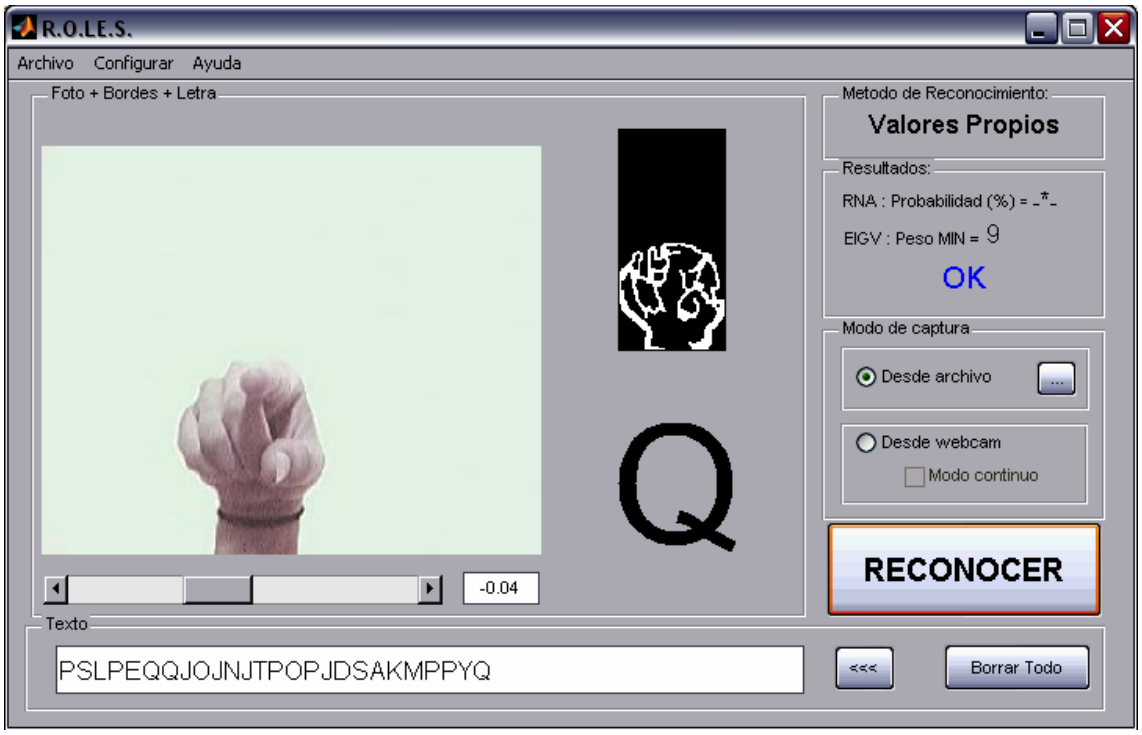


Figura 125 - Resultados carpeta 10

4 MANUAL DE USUARIO PARA LA INTERFAZ GRÁFICA.

PROTOTIPO DE SOFTWARE
PARA EL RECONOCIMIENTO ÓPTICO
DEL ALFABETO INTERNACIONAL DEL LENGUAJE DE SEÑAS.

GUÍA DE USUARIO DE LA INTERFAZ GRÁFICA R.O.LE.S.
(Versión de ROLES 2005.07.20 / Versión del documento 2005.07.21)

CONTENIDOS:

5 Sobre ROLES.

Origen de ROLES.

Las siglas R.O.LE.S. provienen de “Reconocimiento Óptico del Lenguaje de Señas”, aunque la aplicación realiza el reconocimiento de las letras del abecedario solamente. Esta aplicación se desarrolló como un Proyecto Final de Carrera en la Universidad Don Bosco de El Salvador, en el área de Procesamiento Digital de Imágenes. Desarrollado por alumnos de Ingeniería en Electrónica en el año 2005. La aplicación se desarrolló completamente en Matlab 7.04.

Objetivo de ROLES.

El objetivo de la aplicación es desarrollar un prototipo para la implementación de un software que realiza un procedimiento sencillo, basado en algoritmos de reconocimiento

como redes Neuronales Artificiales y Valores Propios. Aunque la aplicación solamente realiza el reconocimiento de las letras del abecedario, puede servir como base teórica para el desarrollo de aplicaciones más avanzadas para el reconocimiento del lenguaje de señas.

Requerimientos mínimos para el uso de ROLES.

a) Computadora con Sistema Operativo Windows® XP.
b) Matlab® 7.04.
c) Webcam USB 1.1 o superior.
d) Si se cumplen los requerimientos de hardware mínimos para el uso de los literales anteriores no habrá inconveniente para la ejecución de la aplicación ROLES.

6 Conociendo la Interfaz Gráfica del programa ROLES.

En ejecutar el programa ROLES, se abrirá una interfaz gráfica como la que se muestra en la figura 1, la cual puede organizarse en 7 áreas:

1 – Área de Menú.

2 – Área de Presentación de Imágenes.

3 – Área de Presentación de Texto.

4 – Área de Presentación del Método de Reconocimiento.

5 – Área de Presentación del Resultado del Reconocimiento.

6 – Área de Controles del Modo de Captura.

7 – Botón para aplicar Secuencia de Reconocimiento.

Figura 126 – Organización de la Interfaz Gráfica.

Por defecto aparecerá una foto de una mano, pero solo como parte de la presentación inicial, el programa aún no está listo para reconocer señas. A continuación se presenta un a breve descripción sobre las 7 áreas mencionadas anteriormente:

1. **ÁREA DE MENÚ**. Presenta 3 menús básicos con opciones de carga de archivos y configuración para diferentes usuarios y enlace a este documento.
   1.1. ***Archivo***. Presenta 3 opciones:
      1.1.1. ***Cargar Red Neuronal***. Para cambiar la red neuronal (previamente entrenada). Al seleccionar esta opción se abrirá una ventana con explorador de archivos que permite al usuario cambiar la red que se usará en la secuencia de reconocimiento. El archivo debe ser de extensión '.mat', puede tener cualquier nombre pero la variable de matlab almacenada dentro del archivo debe llamarse siempre 'red', ya que será tratada con ese nombre en el proceso, de forma que no se produzca una interrupción indeseada de la ejecución del programa por un nombre erróneo.
      1.1.2. ***Cargar Dimensiones***. Cargar un archivo nuevo donde se encuentren las dimensiones de alto y ancho de la mano para las señas de las letras 'a' y la 'b'. Estos archivos pueden ser creados para cada usuario, pero siempre deben guardarse con el nombre 'dimensiones.mat'. Sin embargo los valores cargados por defecto se consideran como promedios, por lo que pueden funcionar para la mayoría de usuarios.
      1.1.3. ***VP interfaz***. Cierra la ventana 'R.O.LE.S' y abre la interfaz para manipulación la base de datos de valores propios para manos.

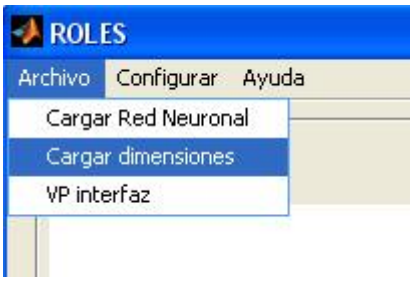


Figura 127 – Menú Archivo

1.2. ***Configurar***. Presenta las siguientes 3 funciones:

1.2.1. ***Método de reconocimiento***. Puede seleccionarse un método combinado entre red neuronal y valores propios o seleccionar uno de ellos individualmente. *Valor por defecto: 'combinado'.*

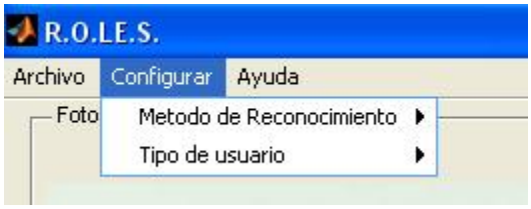


Figura 128 – Menú Configurar

1.2.2. ***Tipo de usuario***. Puede escogerse entre usuario zurdo o diestro. *Valor por defecto: 'diestro'.*

1.3. ***Ayuda***. Presenta información sobre los autores del programa y un enlace para abrir este documento.

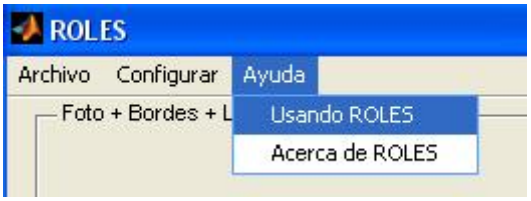


Figura 129 – Menú Ayuda

2. **ÁREA DE PRESENTACIÓN DE IMÁGENES**. En ésta área se mostrará el video en tiempo real capturado por la webcam, la figura del resultado del procesamiento y el caracter obtenido en el reconocimiento, además hay un control deslizante para ajuste de una constante de ajuste de contraste interno.

2.1. ***Ventana de video***. Se presenta el video capturado por la webcam en tiempo real (en vivo).

2.2. ***Ventana de mano esqueletizada y recortada***. Resultado del procesamiento de la imagen capturada, en formato binarizado (2 colores: fondo negro y líneas blancas) con realce de bordes y con recorte de las áreas laterales vacías.
2.3. ***Letra reconocida***. Se presenta el resultado del reconocimiento, según el método seleccionado (red neuronal, valores propios o combinado), mostrando solamente el último carácter del abecedario que resultó de la secuencia de reconocimiento.
2.4. ***Control deslizante***. Consiste de una barra deslizante y un cuadro de texto, ambos ajustan el valor de una constante que es usada por el programa internamente (por lo que no habrá efecto visual en la interfaz gráfica). Si la imagen capturada está extremadamente iluminada el control deslizante debe estar colocado en la extrema izquierda y si la imagen capturada es muy oscura el control debe colocarse en la extrema derecha. Para condiciones intermedias se cuenta con pasos pequeños para su ajuste. El valor puede ser introducido directamente en el cuadro de texto. (El rango para esta constante de ajuste de umbral es: -0.2 < constante < 0.2). El valor por defecto de esta constante al abrir la aplicación es -0.04.

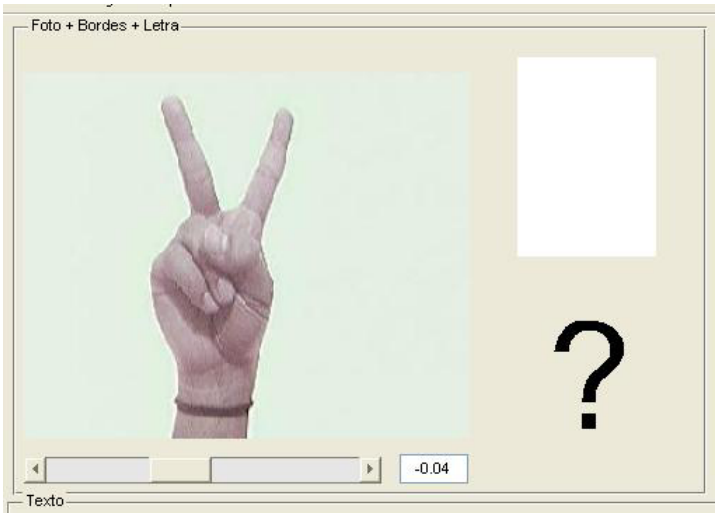


Figura 130 – Área de Presentación de Imágenes

3. **PRESENTACIÓN Y EDICIÓN DE TEXTO**. Espacio para presentar el texto reconocido con 2 botones para borrar un solo carácter o todos al mismo tiempo. También puede copiarse una parte o la totalidad del texto para ser pegado en otro editor.

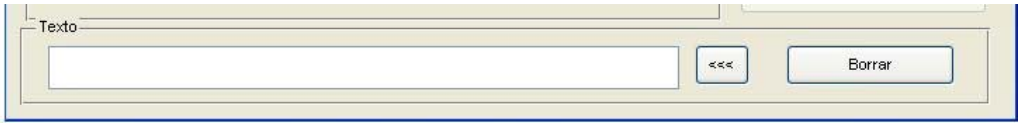


Figura 131 – Área de Presentación de Texto y botones de borrado

3.1. ***Línea de texto***. (Caja blanca) Donde se presentan todos los caracteres del abecedario que han sido reconocidos hasta el momento. El primero en ser reconocido queda a la izquiera y el último que se reconoce se agrega a la extrema derecha.
3.2. ***Borrar un carácter***. Botón con los caracteres "<<<" . Borra el carácter de la extrema izquierda (último que reconoció la aplicación).
3.3. ***Borrar todo el texto***. Botón 'Borrar Todo'. Limpia por completo la línea de texto.

4. **PRESENTACIÓN DE MÉTODO DE RECONOCIMIENTO**. Puede ser 'Red Neuronal', 'Valores Propios' o 'Combinado' (valor por defecto).

Figura 132 – Área de Presentación de Método de Reconocimiento aplicado

5. **PRESENTACIÓN DE RESULTADOS DEL RECONOCIMIENTO**. Se presentan los resultados en 2 constantes básicas:
5.1. Probabilidad de reconocimiento de la red neuronal. Presenta la probabilidad porcentual de que el caracter del abecedario presentado como respuesta corresponda a la seña de la imagen capturada.
5.2. Distancia eigen-manos. Es un valor (peso) relacionado con la distancia del vector obtenido de la imagen reconocida al vector que se tiene como modelo en la base de datos de mano utilizada en el reconocimiento con el método de valores propios.

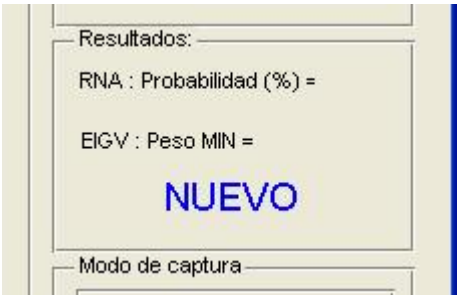


Figura 133 – Área de Presentación de Resultados numéricos

6. **CONTROLES DE MODO DE CAPTURA DE IMAGEN**. Aquí se encuentran los controles con los que se ajusta la forma en que se obtendrán las imágenes que serán procesadas durante el reconocimiento.
6.1. ***Desde archivo***. Al seleccionar este control se habilita el botón localizado a la derecha de este:  , el cual abre una ventana de interfaz gráfica con explorador de

archivos para escoger una imagen previamente almacenada en cualquier unidad de almacenamiento de la computadora. Este control se encuentra activo por defecto al iniciar la aplicación y al habilitar este control se inhabilita automáticamente la opción 'Desde webcam'.

6.2. ***Desde la webcam***. Siempre y cuando esté conectada una webcam, se presentará en tiempo real el video capturado por ésta en la "ventana de video" (descrita en el área de presentación de imágenes /ir/). Este control se encuentra inactivo por defecto al iniciar la aplicación y al habilitar este control se inhabilita automáticamente la opción 'Desde Archivo'.

6.2.1. ***Modo contínuo***. Si está activa al opción de modo de captura desde webcam y si se encuentra una webcam conectada a la computadora, se presentará el video en la ventana respectiva y se realizará el reconocimiento de forma automatizada cada 3 segundos. Este control se encuentra inactivo por defecto al iniciar la aplicación.

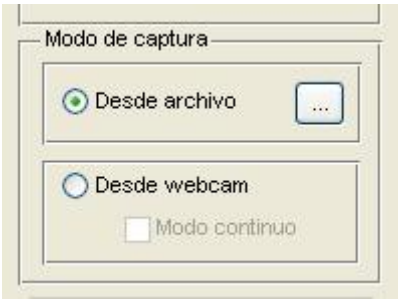


Figura 134 – Área de selección del modo de captura de imágenes

7. **BOTÓN 'RECONOCER'**. Para aplicar la secuencia de Reconocimiento a la imagen actual. Al iniciar la aplicación, este botón no se encuentra habilitado, se activará solamente después de cargar una imagen desde un archivo o cuando se haya seleccionado el modo de captura de imágenes desde la webcam.

Figura 135 – Botón para aplicar secuencia de reconocimiento

7 Formas de aplicar el reconocimiento.

El reconocimiento puede aplicarse a una imagen cargada desde un archivo y también a una imagen capturada mediante una webcam en tiempo real. A continuación se describe la forma de realizar el reconocimiento para cualquiera de las dos posibilidades mencionadas.

7.1 RECONOCIENDO UN IMAGEN CARGADA DESDE UN ARCHIVO.

Primero asegúrese de seleccionar en el Área del Modo de Captura el control “Desde archivo”, como se muestra en la Figura 136. A continuación haga clic en el botón ubicado a la derecha que muestra 3 puntos suspensivos […] para abrir una ventana con explorador de archivos, donde podrá escoger la imagen.

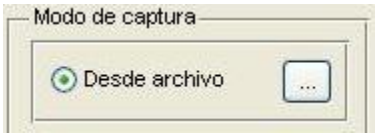


Figura 136 – Modo de captura de imagen desde archivo

(a) Busque la carpeta que contiene la foto (b) Escoja la foto que desea cargar

Figura 137 – Ventana emergente de selección de imagenes a cargar para reconocimiento

Al haber escogido la imagen, haga clic en el botón abrir de la ventana emergente con lo que la imagen aparecerá en la ventana de video de la interfaz gráfica de ROLES, tal como se muestra en laFigura 137.

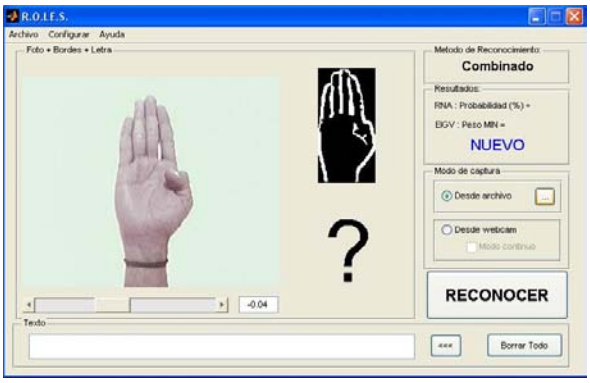


Figura 138 – Imagen cargada para reconocimiento desde archivo

Al cargar la imagen, de forma automática aparecerá la figura de la misma mano en forma esqueletizada, pero aún no se ha realizado el reconocimiento. Para reconocer la seña haga clic en el botón reconocer, después aparecerán los resultados del reconocimiento como en la Figura 139.

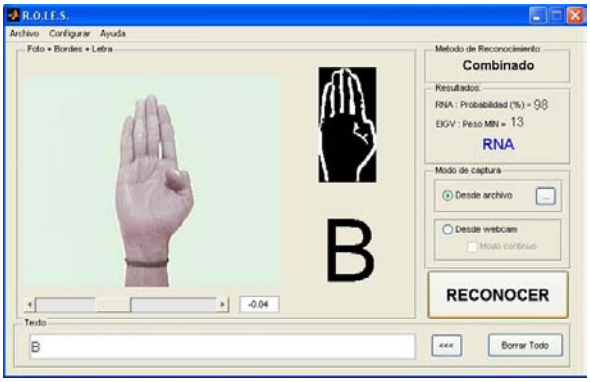


Figura 139 – Imagen cargada desde archivo reconocida

Como resultado de aplicar el reconocimiento, se observa:

En el área de presentación de imágenes, la letra del alfabeto correspondiente a la seña;

En el área de texto se coloca la letra y si ya existieran más en la caja de texto, la última reconocida se agrega a la derecha;

En el área de presentación de resultados, puede encontrarse el porcentaje de probabilidad que se obtuvo con la red neuronal de que la letra presentada corresponda

a la seña de la imagen, también una constante de peso obtenido del reconocimiento utilizando el método de valores propios;
En la misma área de presentación de resultados se observa en forma de texto, el método (valores propios o red neuronal) con el cual se obtuvo mayor probabilidad.

7.2 RECONOCIENDO UNA IMAGEN DESDE EL VIDEO CAPTURADO POR LA WEBCAM.

Primero asegúrese de seleccionar en el Área del Modo de Captura el control “Desde webcam”, como se muestra en la Figura 140.

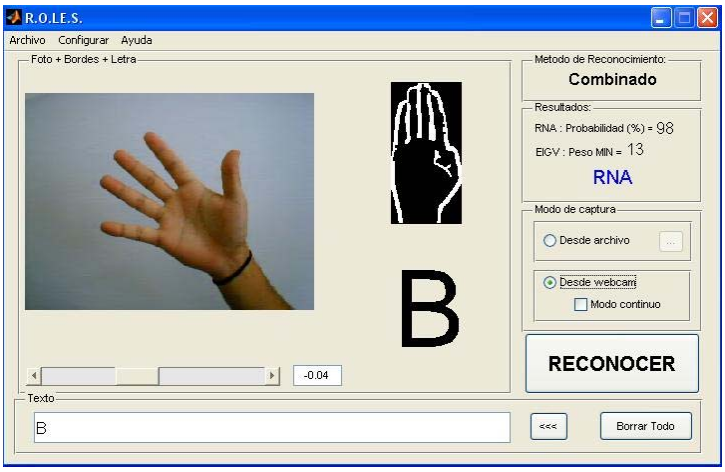


Figura 140 – Imagen tomada desde el video capturado en tiempo real por la webcam

Cuando la mano esté lista (en la posición correspondiente a la seña que desea identificar) haga clic en el botón “RECONOCER”, al hacerlo se actualizará lo siguiente:
La figura de la mano en formato binario (fondo negro y líneas blancas) esqueletizado;
La letra del alfabeto correspondiente a la seña reconocida;
El texto en el área de texto;
Los porcentajes y el peso de valores propios en el área de presentación de resultados.
Estos cambios se muestran en la Figura 141.

(a) Reconocimiento de la letra A desde el video (b) Reconocimiento de la letra C desde el video

Figura 141 – Resultado de reconocimiento de seña desde video capturado por webcam

El área donde se presenta el video no será congelada, siempre mostrará la imagen de la webcam en tiempo real.

Con esta configuración de reconocimiento desde la webcam, para cada seña que desee procesar debe hacer un clic en el botón “RECONOCER”.

7.3 RECONOCIENDO DE FORMA CONTÍNUA DESDE EL VIDEO CAPTURADO POR LA WEBCAM.

Otra posibilidad de reconocimiento que ofrece la interfaz gráfica es reconocer de forma automática las señas del video capturado. En este modo de operación el usuario realizará frente a la webcam las señas que desee identificar mediante la aplicación ROLES. Para configurar este modo de operación debe asegurarse que esté habilitada en el Área del Modo de Captura la opción “Desde webcam” y además activar la marca de selección de la caja de control “Modo contínuo” tal como se ilustra en la Figura 142.

Figura 142 – Selección del modo de captura desde webcam de forma continua (automática)

Cuando la aplicación se encuentra en modo de reconocimiento contínuo desde webcam, el texto y los resultados del reconocimiento se actualizarán de forma ininterrumpida en intervalos de 3 segundos, tiempo suficiente para hacer los cambios de una seña a otra.

La forma de detener este modo de reconocimiento es quitar la marca de la caja de control “Modo contínuo”.

8 APLICACIONES

El reconocimiento óptico de lenguaje de señas R.O.LE.S., está dividido en tres áreas principales, estas son: Captura, Procesamiento de la imagen, Reconocimiento. El área de captura está enfocada en registrar y digitalizar una imagen de la escena para su procesamiento por computadora. El área de procesamiento tiene como finalidad, extraer características de la imagen, es decir, mejorar el contraste, realzar bordes o eliminar ruido o interferencia mediante un filtrado, etc. En el proceso de segmentación, los píxeles de la imagen se agrupan en regiones de interés que tengan un cierto valor o significancia perceptual, para dar paso a la extracción de características. Por último, en el reconocimiento se toma en cuenta la información extraída en la etapa anterior, aquí se realiza la clasificación de las señas, dentro de determinadas clases.

Existen áreas donde se puede aplicar los procesos que anteriormente se mencionan, entre ellas podemos mencionar:

Inspección y Control de Calidad: En ella se puede verificarse el color, las dimensiones y la forma de un objeto, a partir de características extraídas de la imagen, este sería un proyecto de modernización tecnológica que algunas empresas pueden tomar para desarrollarlo en su control de calidad. Por ejemplo, La verificación del estado de un determinado producto, como el monitoreo de calidad de botellas en la cual implique tomar una imagen y luego procesarla, para que al final nos indique la calidad del producto, entre quebrado o bueno, con ello se logra automatizar aun más el proceso de control.

Ensamble y Manufactura: Esta área puede ser aplicada como el sentido de la vista de un sistema de robótica para localizar piezas y ensamblarlas en un equipo; en este caso se requiere el uso de algoritmos que obtengan la posición y orientación tridimensional de un objeto a partir de imágenes, para que esta información pueda transmitirse a un efector robótico.

Control de Tostado del Café: Realizar un control el cual implique obtener datos de un color uniforme que cumpla con el estándar internacional de exportación, para ello es necesario tomar en cuenta lo procesos de captura, procesamiento de la imagen, y reconocimiento.

Reconocimiento de Rostros: Desarrollar un sistema capaz de reconocer rostros de personas, en el cual se extraigan las características más importantes, tomando una gran variedad de imágenes para tener todas las variaciones posibles, y con ello realizar un entrenamiento para obtener buenos resultados en el reconocimiento de la red.

9 CONCLUSIONES

9.1 DEL PROCESO CAPTURA DE LA IMAGEN.

1. Para mejorar la efectividad del reconocimiento, el dispositivo de captura de imágenes (webcam) debe ofrecer características de ajuste automático que permitan obtener una imagen clara aun en condiciones donde no se cuente con abundante iluminación. Las características más importantes que deben ser controladas en la webcam son: Contraste, Brillo y Exposición.

2. Para mejorar la calidad de la imagen capturada y en consecuencia la eficacia del reconocimiento, es necesario encontrar los parámetros de iluminación mínimos que debe cumplir el ambiente donde se realice la toma de fotos con la webcam, ya que de encontrarse en un entorno en el que exista una completa distinción entre el fondo y la mano se obtendrá mayor efectividad.

3. El programa de reconocimiento ha sido probado con fotos adquiridas desde 5 tipos diferentes de webcam, en la gama de $40.00 a $90.00, comprobándose que aunque éstas no sean de las últimas lanzadas al mercado es posible realizar satisfactoriamente el procesamiento de la imagen, sin embargo si no ofrecen nitidez y claridad en la imagen capturada, si influirá directamente la efectividad del reconocimiento, llegando a disminuir en un porcentaje de hasta 50%.

4. Debe realizarse un estudio más detallado para determinar la relación existente ente las dimensiones de la mano, con el propósito de obtener una referencia respecto a la cual puedan definirse todas las dimensiones restantes, tales como longitud máxima y mínima de los dedos, ancho máximo y mínimo de la mano, ancho máximo y mínimo de la muñeca. Al establecer tales relaciones será más fácil identificar cada una de las partes de la mano, permitiendo desarrollar rutinas más efectivas para resaltar determinadas características y eliminar las menos importantes. Ya que

según las pruebas realizadas el área de mayor interés es para la mayoría de las señas donde se encuentran los dedos y su orientación.

5. El programa desarrollado en este trabajo de graduación no realiza la búsqueda del objetivo, por lo que debe ser el usuario quien se asegure de que la mano que será capturada por la webcam ocupe la mayor parte del área de la foto y además que la mano se encuentre en el centro de enfoque de la webcam. La razón por la que no se hizo uso de esta característica es porque algunas webcam ofrecen el enmarcado automático del objetivo aunque algunas veces es deficiente, dejando fuera de marco parte de los dedos e inclusive a veces enfocando áreas que no son objeto de interés.

6. La captura de la imagen debe realizarse a una distancia menor a 1 metro, ya que de hacerlo a mayor distancia se pierde calidad en la imagen capturada y se reduce la efectividad del reconocimiento, por defectos en la extracción y realce de características.

7. Es necesario eliminar cualquier objeto que se encuentre atrás de la mano utilizando un fondo (pantalla) con el fin de aislar elementos que no pertenecen a la mano para que el programa realice las labores de procesamiento rápidamente y con mejores resultados.

8. No es recomendable el uso de guantes para el reconocimiento de señas utilizando el método de extracción de bordes, debido a que el uso de estos provoca que se resalten más líneas que aparecen, originadas por efectos de sombra producidos por los pliegues del guante.

9.2 DEL PROCESAMIENTO DE LA IMAGEN.

9. Es muy difícil determinar los valores idóneos como los umbrales, que se utilizan en procesos de binarización en el procesamiento de la imagen, debido a que deben considerarse una gran variedad de entornos, desde poco iluminados hasta muy iluminados. Por lo que es necesario realizar siempre una configuración según el entorno antes de comenzar a realizar el procesamiento de la imagen y cada vez que estas condiciones de iluminación cambien.

10. Una forma adecuada de uniformizar las imágenes por variaciones en la luminosidad del entorno es aplicar un ecualizado del histograma de la misma. Sin embargo, si en la imagen capturada predominan los colores oscuros en el fondo, el procesado de la imagen hará que la mano se haga más oscura, por lo que este proceso de ecualizado debe realizarse solamente en aquellas imágenes cuyo promedio de intensidad (de la escala de grises) sea muy bajo (fotos muy oscuras), menor a 120 para un rango de 0 a 255..

11. Si el usuario se asegura de que la mano ocupe la mayor parte del área de la foto capturada y que la misma esté centrada, será más fácil que el procesamiento de la imagen elimine aquellos “objetos” que aparecen producto de sombras en el fondo, mejorando sensiblemente el reconocimiento, ya que de lo contrario se corre el riesgo de encontrar medidas erróneas de las dimensiones de la mano, provocando que se agregue un área en la parte superior de la imagen, lo que lleva a un resultado del reconocimiento erróneo.

12. La aparición de nuevos objetos en la imagen capturada producto de la poca iluminación del entorno después de la binarización, reduce sensiblemente la efectividad del reconocimiento porque no permite identificar claramente el área ocupada por la mano obteniendo entonces un recorte defectuoso de la misma.

13. El esqueletizado de la mano puede realizarse con diferentes filtros, obteniéndose los mejores resultados con la aplicación del tipo Canny y resultados considerablemente buenos con Sobel. Sin embargo se considera como mejor alternativa el uso de Canny por la menor cantidad de ruido que produce en la imagen resultante.

14. Es necesario para el recorte de la imagen por la parte de abajo colocar una marca para identificar durante el procesamiento de la imagen desde donde comienza el área de interés, así todas las imágenes procesadas se llevarán a un patrón normalizado, obteniendo mejores resultados para el reconocimiento.

15. Para realizar el reconocimiento de patrones se parte de la idea que para ser comparadas todas las señas, deben llevarse a un patrón normalizado que permita observar las diferencias sustanciales, por lo que resulta imperativo el buen recorte de la imagen para eliminar zonas vacías y de escaso interés (por la escasa o nula variación entre señas).

16. Una forma efectiva de incrementar en más de un 40% la efectividad del reconocimiento de las señas es asegurarse de diferenciar claramente entre las señas realizadas con los dedos extendidos y las señas que se realizan con el puño. La mejor forma de lograrlo es agregar en el caso de las señas con puño un área vacía en la parte superior del mismo. Las dimensiones del área agregada deben ser variables en función de la relación entre la altura máxima de la mano y la altura actual.

9.3 DEL ENTRENAMIENTO DE LAS REDES NEURONALES ARTIFICIALES.

17. El tiempo requerido para el entrenamiento de la red neuronal depende de la cantidad de datos de las entradas, la cantidad de variaciones tomadas para cada categoría, la cantidad de capas ocultas y también del número neuronas de neurona en cada capa, así como de parámetros propios de la red, utilizados para el cálculo de los pesos de cada capa.

18. Para el reconocimiento de imágenes en forma binarizada, las redes de Retropropagación ofrecen buenos resultados, con efectividad en el reconocimiento mayor al 90%, el cual es función de la cantidad de variaciones incluidas durante el entrenamiento y aplicando el reconocimiento a las mismas imágenes con las que se entrenó la red. Cuando se realiza el reconocimiento con imágenes nuevas, esta efectividad se reduce casi en un 40% y si las condiciones luminosas varían drásticamente esta efectividad puede llegar a ser nula.

19. La red de Retropropagación es muy sensible a las variaciones en la imagen como rotación e inclinación de la mano, por lo que la aplicación desarrollada ofrece los mejores resultados cuando las señas se realizan lo más parecido posible a las imágenes con las que se realizó el entrenamiento.

20. Para imágenes binarizadas el uso de la Red Neuronal tipo Hopfield no es sugerido, debido a que los vectores que puede reconocer esta red deben cumplir con el criterio de “Ortogonalidad” y los vectores obtenidos de las señas no lo son.

21. Para realizar el reconocimiento de las imágenes binarizadas mediante una Red Neuronal de tipo Perceptrón, no se obtienen resultados satisfactorios, (son menores al 10% con el tipo de vectores utilizados en este trabajo de investigación), debido a la gran cantidad de neuronas que deben interconectarse y por la sencilla

arquitectura de este tipo de redes, teniendo en cuenta la complejidad en la distribución de la información en los vectores de entrada.

22. La red neuronal ART que se utilizó presenta buenos resultados en la clasificación de las señas, con efectividad en el reconocimiento de aproximadamente 60% para categorías que agrupan varias imágenes de señas muy parecidas entre ellas. Sin embargo los resultados no fueron satisfactorios (efectividad de reconocimiento menor al 25%) para el reconocimiento individual (con la arquitectura desarrollada para las pruebas).

23. El mayor problema para el entrenamiento de la Red ART consistió en encontrar los mejores parámetros de vigilancia y razón de aprendizaje, ya que en la variación de estos recae la buena clasificación de las salidas deseadas.

24. Se decidió utilizar la Red de Retropropagación porque además de identificar las categorías definidas para los grupos de imágenes de señas parecidas, devuelve la probabilidad que cada salida tiene de ser la correcta.

25. Se decidió implementar un nuevo entrenamiento para identificar las categorías reduciendo significativamente la cantidad de datos de entrada, utilizando en lugar de la imagen binarizada los vectores que describen las distancias desde los bordes de la mano hasta los límites de la imagen.

9.4 DEL RECONOCIMIENTO.

26. La utilización de los vectores de distancia desde la mano hasta los límites de la imagen mejoró la efectividad del reconocimiento, incrementándola del 60% al 98% en algunos casos (las imágenes de señas que no se parecen a otras), implementada para la identificación de categorías.

27. Debe buscarse otro tipo de información obtenida a partir de la imagen capturada, la cual debe describir las particularidades de la imagen, las cuales deben ser totalmente diferenciables de las de todas las demás señas.

28. La incorporación del cálculo de los valores propios para cada seña permite obtener otro parámetro de comparación en el caso de aquellas muy parecidas en su forma, ya que estos valores propios ofrecen respuestas muy parecidas cuando se aplican a variaciones de la imagen original, reduciendo la sensibilidad del sistema a la rotación y otros cambios morfológicos producto de variaciones naturales por el cambio de usuario.

29. La mejor forma de realizar el reconocimiento es aplicar un proceso en el cual se combinen los resultados obtenidos por varios métodos.

9.5 DEL ALFABETO INTERNACIONAL DEL LENGUAJE DE SEÑAS.

30. Durante la investigación realizada en este trabajo de graduación se encontraron variaciones en algunas de las formas de representar una letra según el alfabeto internacional del lenguaje de señas, por lo que se decidió usar el obtenido del Ministerio de Educación de El Salvador. Debe definirse claramente cuál es el utilizado en el país para desarrollar la aplicación basada en esa convención.

31. Aunque algunas señas requieren movimiento, como el caso de la “j”, “z” y “ñ”, el proceso de captura, procesamiento y reconocimiento de la imagen utilizado en este trabajo de investigación, hace posible que se definan con una postura fija de la mano, porque el objeto de análisis es siempre una imagen estática.

9.6 DEL INTERFAZ GRÁFICA.

32. A pesar de que Matlab® 7.04 ofrece varios tipos de controles que pueden ser utilizados para la creación de una interfaz gráfica, presenta dificultades en el manejo de imágenes en tiempo real (video) cuando se debe interactuar entre varias ventanas de interfaz gráfica o varias funciones ejecutadas también en tiempo real.

33. La utilización de la interfaz gráfica creada en Matlab® a pesar de su difícil creación presenta grandes facilidades para el manejo de los datos obtenidos y la interacción con las funciones que realizan el procesamiento y reconocimiento de la imagen.

34. Debe considerarse para el desarrollo de aplicaciones similares a ésta, el tiempo requerido por la interfaz para refrescar las imágenes mostradas y los resultados obtenidos, dicho tiempo puede sobrepasar 1 segundo en condiciones de trabajo en las que la computadora tenga muchas aplicaciones en ejecución al mismo tiempo.

10 RECOMENDACIONES

10.1 DEL PROCESO CAPTURA DE LA IMAGEN.

1. La nueva generación de webcam que se encuentran en el mercado, ofrecen habilidades como el seguimiento del objetivo, que pueden ser de alto beneficio para obtener la foto de la mano, siempre y cuando estas cámaras permitan el ajuste de esos parámetros.

2. Desarrollar funciones que permitan cambiar los parámetros de la webcam desde matlab, para realizar una configuración para reducir los efectos de la luminosidad del entorno cada vez que se use la aplicación.

10.2 DEL PROCESAMIENTO DE LA IMAGEN.

3. Debe encontrarse otras fuentes de información que describan la imagen pero que sean constantes aunque la imagen sea deformada o que haya sufrido cambios de escala, cambios en la intensidad de luminosidad.

4. Aplicar un procedimiento automatizado que permita llevar las imágenes a condiciones uniformes sin importar las condiciones del entorno.

10.3 DEL ENTRENAMIENTO DE LAS REDES NEURONALES.

5. Deben realizarse más pruebas que permitan identificar un patrón en la arquitectura de la red, procurando mantener un equilibrio entre efectividad y tiempo de procesamiento.

10.4 DEL ALFABETO INTERNACIONAL DEL LENGUAJE DE SEÑAS.

6. Si se permite la variación en la forma de representar algunas señas del alfabeto internacional, no en su forma sino en orientación, de modo que sea más fácil diferenciarlas entre ellas se puede lograr una reducción significativa en los errores cometidos durante el reconocimiento.

10.5 SOBRE LA ADQUISICIÓN DE IMÁGENES:

7. Existen 2 posibilidades para realizar la adquisición de la imagen con una webcam, haciendo uso de los controladores existentes de windows®. La primera es, hacer uso de unas funciones vfm (video for matlab), la cual es una buena opción, sobre todo cuando se utiliza la versión 6.5 (Release 13) de Matlab. La segunda opción es, en el caso de contar con la versión 7 (Release 14) de Matlab, hacer uso de la caja de herramientas de adquisición de imágenes.

8. La primera opción, ‘vfm’, es muy conveniente, ya que permite la modificación de ciertos parámetros de forma muy directa, con comandos específicos. La segunda, con la caja de herramientas de adquisición de imágenes, hace la captura más rápido, pero la estructura es un poco más compleja.

9. El tipo de webcam, considerando su resolución, formatos de imagen que soporta y precio, puede ser un factor determinante, si no se cumplen ciertos requisitos mínimos, para asegurar cierta calidad mínima de las imágenes.

10. Tanto el fondo (superficie plana ubicada tras la mano que se fotografía) como la luminosidad del espacio donde se realizan las capturas de las imágenes, son factores altamente determinantes en la calidad de la imagen, pues estos dos factores condicionan el resultado de aplicar los procesos de reducción y extracción de características de la imagen. Al tener un fondo claro, como de color blanco, puede mejorarse el contraste entre el fondo y la imagen. Al realizar la captura de imágenes en un entorno con baja luminosidad, estas imágenes resultantes contienen una mayor cantidad de ruido, por efecto de las sombras creadas y la poca cantidad de información, referente a los bordes que puede ser extraída.

10.6 SOBRE EL CAMBIO DE FORMATO DE LAS IMÁGENES.

11. Existen varios formatos de imágenes, el formato ‘jpg’ es uno de los más utilizados, el cual, no ofrece ningún tipo de inconveniente para desarrollar la aplicación tratada en este trabajo de graduación. Este formato guarda la información que describe la imagen en 3 componentes: Luminancia y Crominancia, ésta última en dos componentes más: la de azul y la de rojo. La luminancia es la misma imagen, pero en escala de gris, por lo que resulta fácil obtener esta información. Existen fórmulas matemáticas que pueden ser utilizadas para extraer de las imágenes ‘jpg’, sin embargo, en Matlab se presenta una forma muy fácil y efectiva de lograr esta conversión: el uso de la función rgb2gray.

10.7 SOBRE LA COMPRESIÓN DE DATOS:

12. Una de las formas tomadas en cuenta para reducir la cantidad de datos necesarios para describir una imagen fue el uso de la Transformada discreta del Coseno. Sin embargo, mediante las pruebas experimentales, se concluye, que al menos de la forma en que fue aplicada esta transformación, se logra solamente la reducción de la cantidad de datos necesarios para definir la imagen, sin embargo, no se logra hacer una diferenciación notable entre las imágenes de señas diferentes.

13. El uso de la función `zigzag` se descarta porque no se consideran los píxeles vecinos para la creación del vector de datos que reconocerá la red neuronal. En su lugar se implementa una función llamada `mat2vec` que sí extrae segmentos o submatrices de 9 elementos (3x3) para que estos se ubiquen uno a continuación del otro, concatenando las zonas vecinas de mejor forma. La implementación de este cambio, da como resultado un menor tiempo en el entrenamiento de la red neuronal.

10.8 SOBRE EL RECONOCIMIENTO CON REDES NEURONALES:

14. La selección de la red neuronal para el desarrollo de esta aplicación, no tiene base teórica, el criterio para decidir sobre la funcionalidad de la red en esta aplicación es el porcentaje de acierto obtenido al final. Hasta las últimas pruebas realizadas en esta investigación los resultados son alentadores, obteniéndose alrededor de un 85% de acierto, lo que puede mejorarse notablemente, incrementando las imágenes para cada seña entrenada, o aplicando otro procedimiento a la imagen que extraiga factores descriptores de la misma, que puedan ser utilizados como entradas adicionales para identificar la seña de entrada.

15. Ha sido notable, talvez como desventaja, aunque a bajo nivel, que la red de retro-propagación (o backpropagation) debe ser entrenada con una gran cantidad de variaciones para lograr el reconocimiento con un mayor porcentaje de acierto.

11 BIBLIOGRAFÍA

González, Rafael. Digital Image Processing
Prentice Hall. 2ª Edición.
ISBN 0-20-118075-8
Id. Biblioteca UDB: 621.3 G643 s.f.

Digital image processing using matlab
Gonzalez, Rafael C.
Woods, Richard E, 1954.
Id. Biblioteca UDB: 621.3 G643 2002

Digital Image Processing Algorithms
Pitas, Joannis.
Id. Biblioteca UDB: 621.3 D574 1994

REDES NEURONALES ARTIFICIALES Fundamentos , Modelos y Aplicaciones.
José R. Hilera/ Victor J. Martínez
Id. Biblioteca UDB: 001.535 H644 1995

Rivera, Eduardo. Tesis: Reconocimiento de Patrones Cardíacos utilizando Redes Neuronales.
Universidad Don Bosco. Año 2000
Id. Biblioteca UDB: Tesis 610.28 R621 2000

MATHWORKS, Inc. (The). (2004). Ayuda en pantalla del programa de computadora: MATLAB version 7.0.0.19920 (Release 14), May 06, 2004.

MATHWORKS, Inc. (The). Página web official

http://www.mathworks.com/

http://www.dc.uba.ar/people/eci/2003/segm2.ppt.
Teoría de detección de bordes
Ultima visita: Marzo/2005

http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/apliintarti.htm
Teoría de redes neuronales
Ultima visita: Enero/2005

http://www.control-systems.net/recursos/glosario/
Definición de términos utilizados.
Ultima visita:Febrero/2005

http://www.andygoldstein.org/lapaginadelalumno/materias/contenidos/colordigital/
Teoría sobre imágenes.
Ultima visisa: Enero/2005

http://electronica.com.mx/neural/
Teoría sobre redes neuronales.
Ulima visita: Febrero/2005.

http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/
Teoría sobre redes neuronales.
Ultima visita: Febrero/2005.

http://www.redcientifica.com/doc/doc199903310003.html
Teoría sobre redes neuronales.
Ultima visita: Febrero/2005.

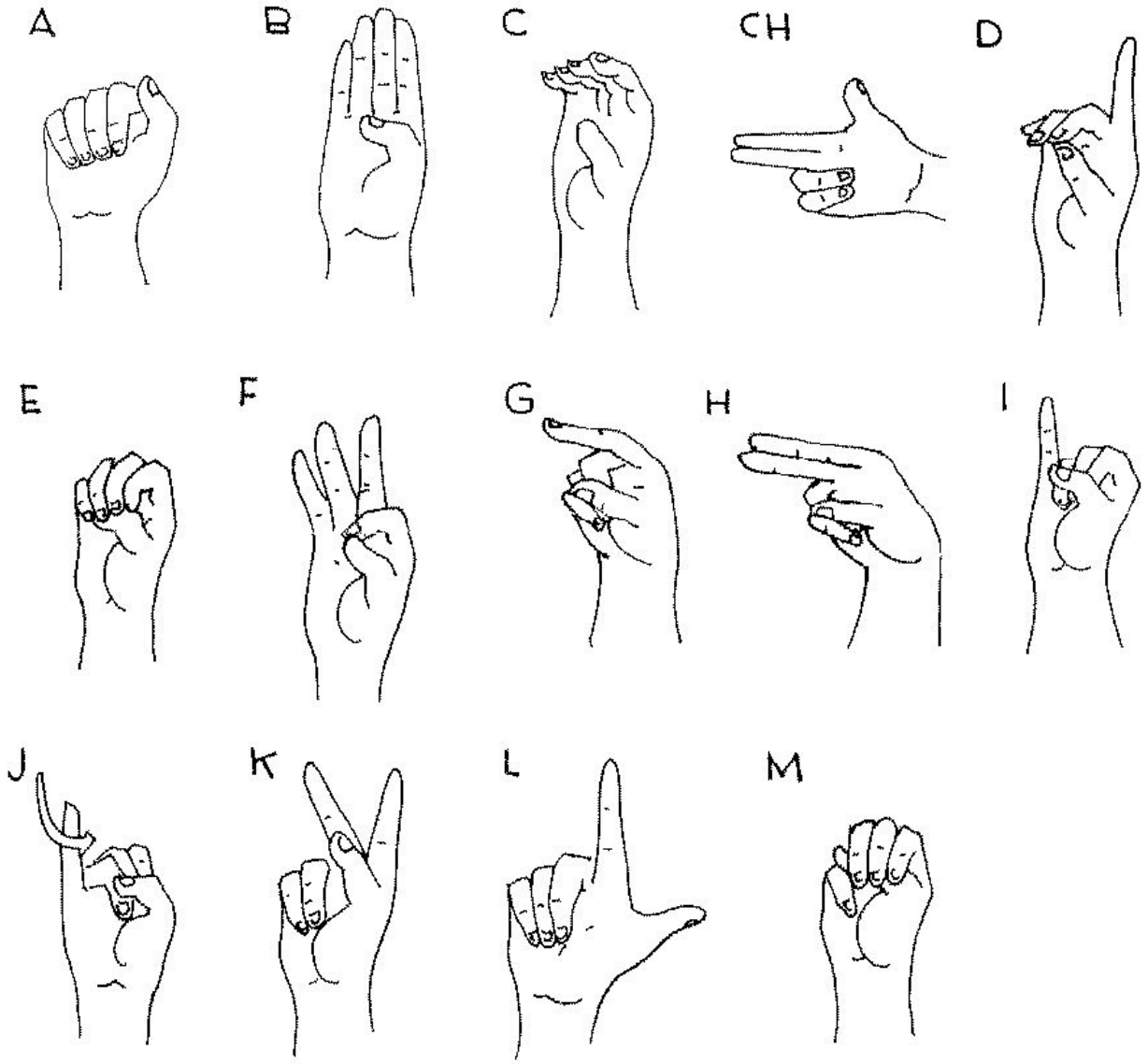
http://ohm.utp.edu.co/neuronales/main.htm

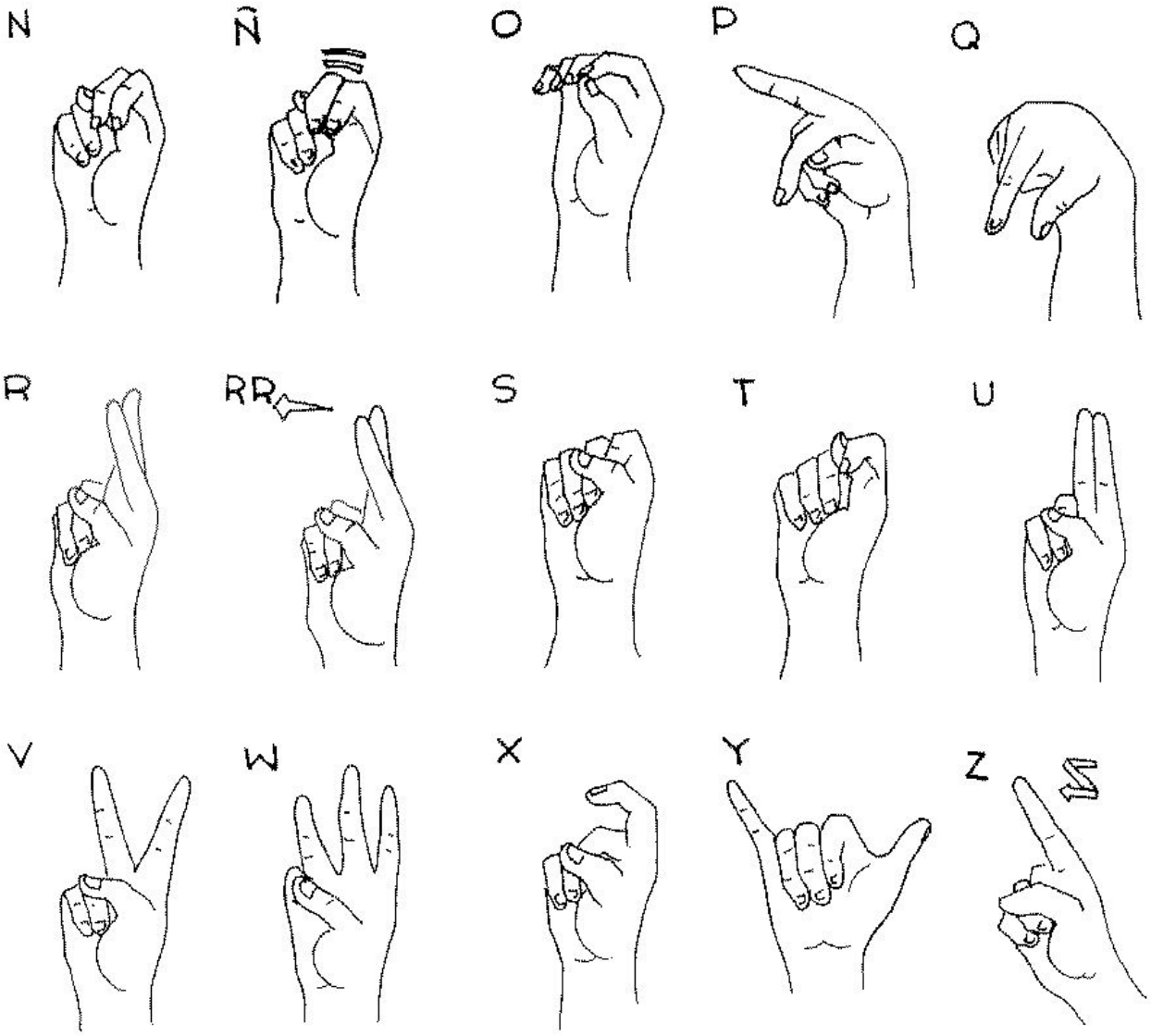
Tutorial de redes neuronales.
Ultima visita: Febrero/2005.

http://www.us.es/gtocoma/pid/pid10/RedesNeuronales.htm#rostro
Diferentes tipos de reconocimientos utilizando redes neuronales.
Ultima visita: Diciembre/2004.

12 ANEXOS

12.1 Alfabeto Internacional / Lenguaje De Sordos.




N
Ñ
O
P
Q
R
RR
S
T
U
V
W
X
Y
Z

12.2 ESPECIFICACIONES TÉCNICAS WEBCAM

Construcción:	Versión del puerto USB: unidad principal remota de la cámara con cable USB.
Almacenamiento de la cámara:	Almacenamiento de lentes en base y en estuche con inclinación manual vertical de 90 grados.
Alimentación:	Desde un puerto USB o desde un concentrador USB con alimentación automática (alimentado con una fuente externa).
Sensor:	Sensor de imágenes CMOS.
Resolución (para modo de video y de imagen fija)	640x480 352x288 320x240 176x144 160x120
Sensibilidad:	6 lux
Formatos de video:	RGB de 24 y 16 bits 4:2:0 YUV planar
Control de exposición:	Automático (control manual opcional con software)
Balance de color	Automático (control manual opcional con software)
Matriz de color:	Predefinida e integral en la cámara.
Campo de visión:	62 grados (horizontal)
Profundidad de campo:	De 75 milímetros hasta el infinito.
Compatibilidad de puerto:	Versión del puerto USB: puerto bus serie universal.
Compatibilidad de software:	Versión del puerto USB: controlador TWAIN, de video para Windows, DirectShow y Still Image que se ejecute en Windows 98, Windows 2000 y Windows me.

12.3 CÓDIGO (ARCHIVOS '.M')

12.3.1 buscafoto.m

```
function ruta_foto = buskfoto()
% Funcion que presenta una ventana (ambiente windows) para buscar un archivo '*.jpg'
% el cual se asigna a la variable 'foto'

[ nombre_arch , nombre_ruta ] = uigetfile( '*.jpg' , 'Buscar foto de mano para procesar (*.jpg)');

if isequal(nombre_arch,0)|isequal(nombre_ruta,0)
   ruta_foto = 0;
else ruta_foto = fullfile(nombre_ruta,nombre_arch);
end
```

12.3.2 buscarna.m

```
function archivo_red = buskrna()

[ nombre_arch , nombre_ruta ] = uigetfile( '*.mat' , 'Seleccione un archivo de red neuronal');

if isequal(nombre_arch,0)|isequal(nombre_ruta,0)
   archivo_red = 0;
else
   archivo_red = fullfile(nombre_ruta,nombre_arch);
end
```

12.3.3 buscarutas.m

```
function folder = buskrutas
% Agregar las carpetas que contienen los archivos '.m'
% usados en el Trabajo de Graduacion
% Recomendacion: Ejecutar cada vez que se inicie matlab.

persistent folder_tg
if not(isdir(folder_tg))
folder_tg='c:\';
end

new_folder_tg = uigetdir(folder_tg,'Buscador de rutas de carpeta');

if new_folder_tg
   folder_tg = new_folder_tg;
   folder = folder_tg;
end
```

12.3.4 bwmorf.m

```
function imagen_bw_modif = bwmorf(imagen_bw)

imagen_bw = bwmorph(imagen_bw,'shrink');
imagen_bw  = bwmorph(imagen_bw,'dilate');
imagen_bw_modif  = bwmorph(imagen_bw,'close');
```

12.3.5 horz.m

```
function [b] = horz( a )

b=a;
r = size(a,1);
c = size(a,2);

for i = 1: r;
   for j =4 : (c-4);
      if a(i,j)==1
         if a(i,j-1)==0 & a(i,j+1)==0 ||  a(i,j-2)==0  & a(i,j+2)==0 || a(i,j-3)==0  & a(i,j+3)==0
            b(i,j)=0;
         end
      end
   end
end
```

12.3.6 proc_foto.m

```
function [ mano, mano_skel, p_A, p_I, p_D ] = proc_foto(img_entrada, a_altura, b_altura, c_luz)
% Función que realiza los procesos de extracción de características a la
% foto de la mano.
% [mano, mano_skel, p_A, p_I, p_D] = proc_foto(foto_mano,a_altura,b_altura)
%
% Argumentos de entrada:
% img_entrada : Foto de la mano capturada con la webcam en formato rgb o en
%               escala de gris.
% a_altura :    Constante de la razón de altura para la letra 'a' respecto
%               al ancho de la muñeca.
% b_altura :    Constante de la razón de altura para la letra 'b' respecto
%               al ancho de la muñeca.
% c_luz :       Constante para compenzar el umbral para la binarización de
%               la imagen de entrada.
%
% Argumentos de salida:
% mano :        foto de la mano en escala de gris recortada y con bordes
%               resaltados en blanco.
% mano_skel :   imagen binarizada de la mano, esqueletizada y recortada,
%               con fondo negro y lineas blancas
% p_A :         Vector de distancias hacia la mano desde arriba.
% p_I :         Vector de distancias hacia la mano desde la izquierda.
% p_D :         Vector de distancias hacia la mano desde la derecha.
try
```

```
if nargin == 4
    % Si la imagen está en formato rgb, será convertida a escala de grises.
    size_foto = size(img_entrada);
    if (numel(size_foto)>2) & (size_foto(3)==3)
        foto_mano = rgb2gray(img_entrada);
    else
        foto_mano = img_entrada;
    end

    % Si la imagen es muy oscura se realiza un equalizado del histograma
    if mean(mean(foto_mano)) < 100
        foto_mano = histeq(foto_mano);
    end

    % Cortando los excesos laterales

    % c_luz = -.04;
    % La constante 'c_luz' es un valor utilizado para afectar el umbral de
    % binarización. Si la imagen es clara 'c_luz' toma un valor negativo y si
    % la imagen es oscura 'c_luz' debe tener un valor positivo. En ambos
    % casos el rango de 'c_luz' debe ser: 0 < abs(c_luz) < 0.1

    % Binarización de la mano
%     foto_bw = im2bw(foto_mano,.85-c_luz); % Para imágenes claras (de las carpetas,
    % % resultó ser un buen valor.
    foto_bw = im2bw(foto_mano,graythresh(foto_mano)-c_luz);

    % Eliminando agujeros internos en la imagen binarizada:
    foto_bw2 = bwmorph(not(foto_bw),'dilate');
    foto_2bw = bwareaopen(imfill(foto_bw2,'holes'),fix(.3*bwarea(foto_bw2)));
    foto_2bw = bwmorph(foto_2bw,'dilate');
    % foto_2bw = imclose(foto_2bw,strel('disk',25,8));

    % Suavizando la foto en escala de gris aplicando opening con un elemento
    % estructural en forma de diamante:
    se = strel('diamond',5);
    foto_go = imopen(foto_mano,se);

    % Eliminación del fondo en la imagen en escala de gris:
    foto_mano = foto_go.*uint8(foto_2bw);

    % Encontrando los centroides de la imagen binarizada:
    L = bwlabel(foto_2bw);
    s = regionprops(L, 'centroid');
    centroides = cat(1, s.Centroid);
    % Selección del centroide más cercano al centro de la imagen:
    % (Debe asumirse que el usuario que haga uso del programa debe procurar que
    % la mano ocupe la mayor parte de la foto así como también que se encuentre
    % lo más centrada posible)
    if size(centroides,1)>1
        for par = 1:size(centroides,1)
            if abs(centroides(par,1) - (size(foto_2bw,2)/2)) < 0.12*size(foto_2bw,2)
                centroide = fix(centroides(par,:));
            end
```

```
    end
else
    centroide = fix(centroides);
end

% Cortando la mano hasta los bordes externos a partir del centroide
% encontrado:
% Encontrando el área ocupada por la mano en el eje horizontal:
zonas_W_E = find(any(foto_2bw));
% Encontrando el límite izquierdo de la mano:
pos_izquierdo = find( zonas_W_E < centroide(1),1,'first');
izquierdo = zonas_W_E(pos_izquierdo);
if izquierdo < 2
    izquierdo = 2;
end
% Encontrando el límite derecho de la mano:
pos_derecho = find( zonas_W_E > centroide(1),1,'last');
derecho = zonas_W_E(pos_derecho);
if derecho > size_foto(2)-2;
    derecho = size_foto(2)-2;
end
% Eliminación de las áreas laterales del fondo:
mano = foto_mano(:,izquierdo-1:derecho+1);
mano_2bw = foto_2bw(:,izquierdo-1:derecho+1);

% Encontrando el área ocupada por la mano en el eje vertical:
mano_cany = edge(mano,'canny',0.14,.95);

arriba = find(any(mano_2bw,2),1,'first');
if arriba < 2
    arriba = 2;
end
% Eliminando área vacía del fondo por la parte de arriba:
mano = mano(arriba-1:end,:);
mano_cany = mano_cany(arriba-1:end,:);
mano_2bw = mano_2bw(arriba-1:end,:);

% Recorte de la mano por la parte de abajo:

% Realce de las líneas horizontales en la mano, para dejar la línea de la
% marca utilizada en la muñeca:
mano_h = horz(mano_cany);


% Calculando límites verticales de la mano (arriba y abajo):
arriba = find(any(mano_cany,2),1,'first');
abajo = find(bwareaopen(any(mano_h,2),2),1,'last');

% Calculando el ancho de la muñeca a partir del corte de la parte más baja
% de la mano:
muneca = mano_cany(end-20:end,:);
muneca = bwmorf(muneca);
muneca_v = vert(muneca);
muneca_1_h = find(any(muneca_v));
```

```
ancho_muneca = muneca_1_h(end) - muneca_1_h(1);

% Comprobando que la ubicación del límite inferior (abajo) no produzca por
% fallas en la detección una altura menor al mínimo estimado:
if (abajo-arriba) < .9*(a_altura * ancho_muneca)
   if (a_altura * ancho_muneca) < size(mano_cany,1)
      abajo = fix(arriba + (a_altura * ancho_muneca));
   else
      abajo = fix(size(mano_cany,1)-5); % Estaba restando 5
   end
end

% Recorte de imagenes hasta el límite inferior:
mano = mano(1:abajo,:);
mano_cany = mano_cany(1:(abajo),:);
mano_skel = bwmorf(mano_cany);
mano_skel = mano_skel(1:abajo,:);
mano_2bw = mano_2bw(1:abajo,:);

% Cálculo de la altura de la mano:
altura = abajo-arriba;

% Agregando área en blanco para manos de puño, decidiendo el tamaño a
% partir de la altura calculada y los valores standar de argumentos de
% entrada:
espacio = fix((b_altura * ancho_muneca) - altura);
tam_2bw = size(mano_2bw,2);
area_0 = zeros([espacio,tam_2bw]);

% Agregando el área en blanco a las imágenes
mano = [ uint8(area_0) ; mano ];
mano_skel = [ logical(area_0) ; mano_skel ];
mano_2bw = [ logical(area_0) ; mano_2bw ];

% Redimensionando la imagen mano binarizada a tamaño standar para
% extracción de vectores de perfiles:
mano_2bw = imresize(mano_2bw,[90,60],'bicubic');

% Encontrando los perfiles a partir de la imagen de area de la mano.
% (Procedimiento realizado fila por fila y columna por columna):
[nf,nc] = size(mano_2bw);
for fila = 1:nf
   if any(mano_2bw(fila,:))
      p_izq(fila) = find(mano_2bw(fila,:),1,'first');
      p_der(fila) = find(mano_2bw(fila,:),1,'last');
   else
      p_izq(fila) = nc;
      p_der(fila) = 1;
   end
end
for columna = 1:nc
   if any(mano_2bw(:,columna))
      p_arr(columna) = find(mano_2bw(:,columna),1,'first');
   else
      p_arr(columna) = nf;
```

```
        end
    end

    % Normailzación de los vectores de perfiles encontrados para contener
    % valores en el rango [0 1]:
    p_A = p_arr./90;
    p_I = p_izq./60;
    p_D = p_der./60;

else
    help proc_foto
    error('Faltan argumentos')
end
catch
    mano=[];
    mano_skel=[]; p_A=[]; p_I=[]; p_D=[];
end
```

12.3.7 reconocer.m

```
function letra = reconocer(red,data_mano)
% Función para realizar la simulación de una red con una imagen de prueba
% letra = reconocer(data_mano)

if nargin == 0
    help reconocer;
    error('No hubo entrada')
else
    salidas = sim(red,data_mano);
    letras = 'abcdefghijklmnopqrstuvwxyz';
    posicion = find(salidas==max(salidas));
    letra = letras(posicion);
end
```

12.3.8 reconocer_eigv.m

```
function salida = reconocer_eigv(data_mano)

entrada=data_mano;

if (exist('manos_database.dat')==2)
    load('manos_database.dat','-mat');
    % manos_cantidad es el número de caras en la database.
    % Turk's paper: "M"
    % Estas imagenes se agrupan en clases.
    % Cada clase debería incluir un número de imagenes por cada
    % persona, con algunas variaciones de la seña o la claridad
    % (luz) de la imagen.
    matriz_ce = zeros(size(data{1,1},1),manos_cantidad);
    for ii=1:manos_cantidad
        matriz_ce(:,ii)=double(data{ii,1});
```

```
end
suma_s=sum(matriz_ce,2);
media=suma_s/manos_cantidad;
for ii=1:manos_cantidad
   matriz_ce(:,ii)=matriz_ce(:,ii)-media;
end
matriz_ce=matriz_ce/sqrt(manos_cantidad);
% "matriz_ce" es Matriz "A" de Turk's paper
matriz_L=matriz_ce'*matriz_ce;
% matriz "matriz_L" is matriz "L" de Turk's paper

% eigenvalues y eigenvectors de la matriz "reducida" A'*A
[V,D] = eig(matriz_L);
% La siguiente multiplicación se implementa para obtener
% eigenvectors de la matriz original A*A'
Vtrue=matriz_ce*V;
Dtrue=diag(D);

% Los eigenvalues son ordenados y solo se toma de ellos M'.
% Se define M' igual al número de clases
% (clases_maximo-1)
[Dtrue,matriz_ordenada]=sort(Dtrue);
Dtrue=flipud(Dtrue);
matriz_ordenada=flipud(matriz_ordenada);
Vtrue(:,1:manos_cantidad)=Vtrue(:,matriz_ordenada);

Vtrue=Vtrue(:,1:clases_maximo-1);
Dtrue=Dtrue(1:clases_maximo-1);

% Se calculan los componentes eigen de la entrada
% normalizada (ajustada al promedio)
peso=Vtrue'*(entrada-media);

peso_database=zeros(clases_maximo-1,clases_maximo-1);
numero_elemento_clase=zeros(clases_maximo-1,1);
for ii=1:manos_cantidad
   ingresso_database=double(data{ii,1});
   clase_database=data{ii,2};
   peso_actual=Vtrue'*(ingresso_database-media);
   peso_database(:,clase_database)=peso_database(:,clase_database)+peso_actual;
   numero_elemento_clase(clase_database)=numero_elemento_clase(clase_database)+1;
end
for ii=1:(clases_maximo-1)
   peso_database_media(:,ii)=peso_database(:,ii)/numero_elemento_clase(ii);
end
% "peso_database_media" es una matriz con los componentes
% eigen promediados de las imagenes presentes en database.
% Cada clase tiene su valor eigen promediado.
% Se desea encontrar el vector (normal) más cercano al
% componente eigen de entrada.

distancia_peso=zeros(clases_maximo-1,1);
for ii=1:(clases_maximo-1)
   distancia_peso(ii)=norm(peso-peso_database_media(:,ii));
end
```

```
    [peso_minimo,posicion_peso_minimo]=min(distancia_peso);
    % Se evalúa la distancia de la imagen de entrada
    % media-normalizada para determinar si la imagen de entrada
    % es una seña o no
    proyeccion=zeros(size(data{1,1},1),1);
    for ii=1:(clases_maximo-1)
       proyeccion=proyeccion+peso(ii)*Vtrue(:,ii);
    end
    distancia_spacio_facc=norm((entrada-media)-proyeccion);

    dist=num2str(peso_minimo);
    dist_spac=num2str(distancia_spacio_facc);

%     mensaje1=strcat('La clase más cercana es la:_', num2str(posicion_peso_minimo));
%     mensaje2=strcat('                (dist_=_',dist(1:5), ')_;');
%     mensaje3=strcat('                (dist desde espacio imagen =_', dist_spac(1:5) , ')');
%
%     msgbox(strcat(mensaje1,mensaje2,mensaje3),'Resultado de Reconocimiento','help');

    letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    posicion = posicion_peso_minimo;
    letra = letras(posicion);

    salida = [letra , peso_minimo , distancia_spacio_facc ];

else
    warndlg('No es posible el procesamiento de la imagen. Database vacía.','Advertencia')
end
```

12.3.9 reconocer_red.m

```
function  [letras_posibles , probabilidades ] = reconocer_red( red , data_mano)
% Función para realizar la simulación de una red con una imagen de prueba
% letra = reconocer(data_mano)

if nargin == 0
    help reconocer;
    error('No hubo entrada')
else
    salidas = sim(red,data_mano);
    letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    [prob1 , pos1]= max(salidas);
    salidas(pos1) = 0;
    [prob2 , pos2] = max(salidas);
    salidas(pos2) = 0;
    [prob3 , pos3] = max(salidas);

    letras_posibles = [ letras(pos1) , letras(pos2) , letras(pos3) ];
    probabilidades = [ prob1 , prob2 , prob3 ];
end
```

12.3.10 reconocer_video.m

```
function reconocer_video

try
handles=guidata(findobj('Tag','roles'));

handles.imagen=getsnapshot(handles.obj);
if handles.usuario_zurdo
   imagen=fliplr(rgb2gray(handles.imagen));
end
% handles.imagen=imagen;
imag = handles.imagen;
a_h = handles.dimensiones.a_altura;
b_h = handles.dimensiones.b_altura;
umb = handles.umbral;

[ mano, skel, p_A, p_I, p_D ] = proc_foto(imag, a_h, b_h, umb);

if isempty(mano)
   return
end

handles.imagen_bw = skel;
handles.data_mano = [p_I'; p_A'; p_D'];

axes(handles.bordes_axe)
% set(gcf,'NextPlot','add'); % Por compatibilidad con matlab 7.01
imshow(handles.imagen_bw);

if handles.bandera_metodo
   metodo_opc = handles.bandera_metodo;
   switch metodo_opc
      case 2
         % Método de Reconocimiento con Eigenvalues
         skel_p = reduc_img(handles.imagen_bw,1500);
         skel_v = skel_p(:);
         if length(skel_v)<1550
            cola = zeros(1550-length(skel_v),1);
         else
            cola = zeros(50,1);
         end
         img_1 = [skel_v ; cola ];
         entrada = img_1(1:1550,1);
         salida = reconocer_eigv(entrada);
%           [letra_vp , peso_minimo , distancia_spacio_facc ] = salida;
         letra = salida(1);
         peso_min = salida(2);
         set(handles.eigv_peso_lbl,'String',int2str(peso_min));
         set(handles.probab_lbl,'String','-*-');
         if peso_min > 20
            set(handles.resultado_lbl,'String','PESO > 20');
         else
            set(handles.resultado_lbl,'String','OK');
```

```
        end
      case 1
        % Método de Reconocimiento con Red Neuronal
        red = handles.red_neuronal;
        vec_mano = handles.data_mano;
%           respuesta = reconocer_red(red,vec_mano);
        [letras_rna, probs_rna] = reconocer_red(red,vec_mano);
        letra = letras_rna(1);
        probs = fix(1000.*probs_rna)./10;
        porcentaje = probs(1);
        set(handles.probab_lbl,'String',int2str(porcentaje));
        set(handles.eigv_peso_lbl,'String','-*-');
        if porcentaje < 90
          set(handles.resultado_lbl,'String','NO COINCIDE');
        else
          set(handles.resultado_lbl,'String','OK');
        end
  end
else
  % Método de Reconocimiento combinado
  % ----------------------------------

  % En primer lugar se encuentra el resultado de la red neuronal:
  red = handles.red_neuronal;
  vec_mano = handles.data_mano;
%   salida_rna = reconocer_red(red,vec_mano);
  [ letras_rna, probs_rna] = reconocer_red(red,vec_mano);
%   letras_rna = salida_rna(1:3);
  probs = fix(1000.*probs_rna)./10;
  % En segundo lugar se encuentra el resultado con eigen-vectors:
  skel_p = reduc_img(handles.imagen_bw,1500);
  skel_v = skel_p(:);
  if length(skel_v)<1550
    cola = zeros(1550-length(skel_v),1);
  else
    cola = zeros(50,1);
  end
  img_1 = [skel_v ; cola ];
  entrada = img_1(1:1550,1);
  salida_eigv = reconocer_eigv(entrada);
  letra_eigv = salida_eigv(1);

  pos_letra = find( letras_rna == salida_eigv );
  peso_min = salida_eigv(2);
  set(handles.eigv_peso_lbl,'String',int2str(peso_min));

  guidata(gcf,handles)    % Como en reconocer_ciclo2
  handles=guidata(gcf);   % Como en reconocer_ciclo2

  if probs(1) < 95
    if pos_letra
      porcentaje = probs(pos_letra);
      set(handles.probab_lbl,'String',int2str(salida_eigv(2)));
      set(handles.resultado_lbl,'String','COINCIDENCIA');
      letra = salida_rna(pos_letra);
```

```
        else
            letra = letra_eigv;
            porcentaje = probs(1);
            set(handles.probab_lbl,'String',int2str(porcentaje));
            set(handles.resultado_lbl,'String',' EIGV ');
        end
    else
        % El porcentaje obtenido del reconocimiento de la RNA es
        % suficientemente alto ( probabilidad >= 0.95 ).
        letra = letras_rna(1);
        porcentaje = probs(1);
        set(handles.probab_lbl,'String',int2str(porcentaje));
        set(handles.resultado_lbl,'String',' RNA ');
    end
end

handles.letra=letra;
set(handles.letra_lbl,'String',letra);
texto_actual = get(handles.texto_txt,'String');
texto_nuevo = [texto_actual letra];
set(handles.texto_txt,'String',texto_nuevo);

guidata(gcf, handles);
catch
    return
end
```

12.3.11 reduc_img.m

```
function img_peq = reduc_img(imagen,long)
% Reducir tamaño de imagen usando interpolacion "bilinear"
% Argumentos:
% img : imagen binarizada de entrada
% long : cantidad de pixeles en la imagen de salida
% img_peq : imagen binarizada redimensionada

[tf tc] = size(imagen);
razon_compres = sqrt(long/(tf*tc));
fc = ceil([tf tc].*razon_compres);
fc3 = fc - mod(fc,3);

img_peq = imresize(imagen,fc3,'bicubic');
```

12.3.12 roles.m

```
function varargout = roles(varargin)
% ROLES M-file for roles.fig
%      ROLES, by itself, creates a new ROLES or raises the existing
%      singleton*.
%
%      H = ROLES returns the handle to a new ROLES or the handle to
```

```
%      the existing singleton*.
%
%      ROLES('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ROLES.M with the given input arguments.
%
%      ROLES('Property','Value',...) creates a new ROLES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before roles_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to roles_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help roles

% Last Modified by GUIDE v2.5 24-Aug-2005 22:37:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @roles_OpeningFcn, ...
   'gui_OutputFcn',  @roles_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before roles is made visible.
function roles_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to roles (see VARARGIN)

% Choose default command line output for roles

handles.output = hObject;
axes(handles.imagen_axe);
% set(gcf,'NextPlot','add');    % Por compatibilidad con Matlab 7.0.1
```

```
handles.imagen = imread('v.jpg');
handles.himagen = imshow(handles.imagen);

handles.umbral=-.04;
handles.umbral_max=.19;
handles.umbral_min=-.2;
handles.step_grande=.1;
handles.step_peque=.02;

set(handles.umbral_sld,'Max',handles.umbral_max,'Min',handles.umbral_min)
set(handles.umbral_sld,'SliderStep',[handles.step_peque,handles.step_grande]...
    /(handles.umbral_max-handles.umbral_min))

axes(handles.bordes_axe);
% set(gcf,'NextPlot','add');    % Por compatibilidad con Matlab 7.0.1
handles.imagen_bw = ones(200,140);
% handles.himagen = imshow(handles.imagen_bw);
imshow(handles.imagen_bw)

handles.t=[];

load('red','-mat');
handles.red_neuronal = red;
handles.dimensiones = load('dimensiones');

handles.ruta = 'c:\';

% set(handles.bordes_axe,'UserData','bordes')

guidata(hObject, handles);

set(handles.umbral_sld,'Value',handles.umbral)


% Condiciones por defecto al iniciar:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Método de reconocimiento por defecto: Combinado
handles.bandera_metodo = 0;
% Método de obtener imagen
set(handles.config_user_derecho_mnu,'Checked','on')
set(handles.config_user_zurdo_mnu,'Checked','off')
% Tipo de usuario por defecto: Diestro
handles.usuario_zurdo = 0;

set(0,'ShowHiddenHandles','on')
% Update handles structure

handles=umbral_sld_Callback(hObject, eventdata, handles);
guidata(hObject, handles);


% UIWAIT makes roles wait for user response (see UIRESUME)
% uiwait(handles.roles);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = roles_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in abrir_btn.
function abrir_btn_Callback(hObject, eventdata, handles)
% hObject    handle to abrir_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[ nombre_arch , nombre_ruta ] = uigetfile( '*.jpg' , ...
   'Buscar foto de mano para procesar (*.jpg)', ...
   handles.ruta);
if isequal(nombre_arch,0)|isequal(nombre_ruta,0)
   return
else handles.ruta = fullfile(nombre_ruta,nombre_arch);
end

handles.imagen=imread(handles.ruta);
guidata(hObject, handles);
handles=procesar_imagen(hObject, eventdata, handles);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function texto_txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to texto_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function texto_txt_Callback(hObject, eventdata, handles)
% hObject    handle to texto_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of texto_txt as text
%        str2double(get(hObject,'String')) returns contents of texto_txt as a double

% --- Executes on button press in backspace_btn.
function backspace_btn_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to backspace_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Borrado del último caracter o letra del texto actual.
texto_actual = get(handles.texto_txt,'String');
texto_nuevo = texto_actual(1:end-1);
set(handles.texto_txt,'String',texto_nuevo);

guidata(hObject, handles);

% --- Executes on button press in borrar_btn.
function borrar_btn_Callback(hObject, eventdata, handles)
% hObject    handle to borrar_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Borrado de todo el texto.
set(handles.texto_txt,'String','')

guidata(hObject, handles);

% --- Executes on button press in reconocer_btn.
function reconocer_btn_Callback(hObject, eventdata, handles)
% hObject    handle to reconocer_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if get(handles.webcam_rbt,'Value')
   if ~get(handles.continuo_ckb,'Value')
      imagen=getsnapshot(handles.obj);
      if handles.usuario_zurdo
         imagen=fliplr(rgb2gray(imagen));
      end
      handles.imagen=imagen;

      imag = handles.imagen;
      a_h = handles.dimensiones.a_altura;
      b_h = handles.dimensiones.b_altura;
      umb = handles.umbral;

      [ mano, skel, p_A, p_I, p_D ] = proc_foto(imag, a_h, b_h, umb);
      if isempty(mano)
         return
      end
      handles.imagen_bw = skel;
      handles.data_mano = [p_I'; p_A'; p_D'];
      % Actualizando los datos obtenidos
      guidata(hObject, handles);

      axes(handles.bordes_axe)
      %        set(gcf,'NextPlot','add');
      imshow(handles.imagen_bw);
   else
```

```
        return
    end
end

if handles.bandera_metodo
    metodo_opc = handles.bandera_metodo;
    switch metodo_opc
        case 2
            % Método de Reconocimiento con Eigenvalues
            skel_p = reduc_img(handles.imagen_bw,1500);
            skel_v = skel_p(:);
            if length(skel_v)<1550
                cola = zeros(1550-length(skel_v),1);
            else
                cola = zeros(50,1);
            end
            img_1 = [skel_v ; cola ];
            entrada = img_1(1:1550,1);
            salida = reconocer_eigv(entrada);
            % salida = [letra_vp , peso_minimo , distancia_spacio_facc ];
            letra = salida(1);
            peso_min = salida(2);
            set(handles.eigv_peso_lbl,'String',int2str(peso_min));
            set(handles.probab_lbl,'String','-*-');
            if peso_min > 20
                set(handles.resultado_lbl,'String','PESO > 20');
            else
                set(handles.resultado_lbl,'String','OK');
            end
        case 1
            % Método de Reconocimiento con Red Neuronal
            red = handles.red_neuronal;
            vec_mano = handles.data_mano;
%           respuesta = reconocer_red(red,vec_mano);
            [letras_rna, probs_rna] = reconocer_red(red,vec_mano);
            letra = letras_rna(1);
            probs = fix(1000.*probs_rna)./10;
            porcentaje = probs(1);
            set(handles.probab_lbl,'String',int2str(porcentaje));
            set(handles.eigv_peso_lbl,'String','-*-');
            if porcentaje < 90
                set(handles.resultado_lbl,'String','NO COINCIDE');
            else
                set(handles.resultado_lbl,'String','OK');
            end
    end
else
    % Método de Reconocimiento combinado
    % ---------------------------------

    % En primer lugar se encuentra el resultado de la red neuronal:
    red = handles.red_neuronal;
    vec_mano = handles.data_mano;
%     salida_rna = reconocer_red(red,vec_mano);
    [letras_rna, probs_rna] = reconocer_red(red,vec_mano);
%     letras_rna = salida_rna(1:3);
```

```
    probs = fix(1000.*probs_rna)./10;

    % En segundo lugar se encuentra el resultado con eigen-vectors:
    skel_p = reduc_img(handles.imagen_bw,1500);
    skel_v = skel_p(:);
    if length(skel_v)<1550
       cola = zeros(1550-length(skel_v),1);
    else
       cola = zeros(50,1);
    end
    img_1 = [skel_v ; cola ];
    entrada = img_1(1:1550,1);  % Para el caso de que el vector contenga más  de 1550 elementos
    salida_eigv = reconocer_eigv(entrada);
    letra_eigv = salida_eigv(1);

    pos_letra = find( letras_rna == letra_eigv );
    peso_min = salida_eigv(2);
    set(handles.eigv_peso_lbl,'String',int2str(peso_min));

    if probs(1) < 95
       if pos_letra
          porcentaje = probs(pos_letra);
          set(handles.probab_lbl,'String',int2str(salida_eigv(2)));
          set(handles.resultado_lbl,'String','COINCIDENCIA');
          letra = letras_rna(pos_letra);
       else
          letra = letra_eigv;
          porcentaje = probs(1);
          set(handles.probab_lbl,'String',int2str(porcentaje));
          set(handles.resultado_lbl,'String',' EIGV ');
       end
    else
       % El porcentaje obtenido del reconocimiento de la RNA es
       % suficientemente alto ( probabilidad >= 0.95 ).
       letra = letras_rna(1);
       porcentaje = probs(1);
       set(handles.probab_lbl,'String',int2str(porcentaje));
       set(handles.resultado_lbl,'String',' RNA ');
    end
end

handles.letra=letra;
set(handles.letra_lbl,'String',letra);
texto_actual = get(handles.texto_txt,'String');
texto_nuevo = [texto_actual letra];
set(handles.texto_txt,'String',texto_nuevo);

guidata(hObject, handles);

% --------------------------------------------------------------------
function roles_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to roles_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function roles_carga_red_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to roles_carga_red_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Abriendo el archivo de la red Backpropagation:

archivo_red = buskrna;
if ~archivo_red
   return
end

load(archivo_red,'red');
handles.red_neuronal=red;

handles.bandera_metodo = 1;

guidata(hObject, handles);


% --------------------------------------------------------------------
function roles_carga_dim_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to roles_carga_dim_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Escogiendo el archivo de dimensiones de la mano:
archivo_dim = buskdim;
if archivo_dim
   %     dimensiones = load(archivo_dim,'dimensiones');
   handles.dimensiones = load(archivo_dim);
   guidata(hObject, handles);
end

guidata(hObject, handles);


% --------------------------------------------------------------------
function roles_carga_bdmanos_Callback(hObject, eventdata, handles)
% hObject    handle to roles_carga_bdmanos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Abriendo el archivo .m que contiene realiza las tareas de creación y
% edición de la bse de datos, usando eigenvalues (eig_manos.m)

% Hay problema al abrirlo, porque cierra la GUI
eig_manos
handles.bandera_metodo = 0;

guidata(hObject, handles);
```

```
% --------------------------------------------------------------------
function configurar_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to configurar_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function ayuda_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function ayuda_version_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda_version_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

L0='UNIVERSIDAD DON BOSCO       (AGOSTO / 2005)                    ';
L1='Reconocimiento Optico del Lenguaje de Señas                    ';
L2='Autores:                                                       ';
L3='[+] Mario E. Orellana Crespin                                  ';
L4='[+] Juan C. Rivera García                                      ';
L5='[+] J. Armando Fonseca Rosales                                 ';
L6='-------------------------------------------------------------------------';
L7='Version: 2005.08.20';
msgbox([L0 L1 L2 L3 L4 L5 L6 L7],'R.O.LE.S.','help')

% --------------------------------------------------------------------
function ayuda_descripcion_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda_descripcion_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% msgbox('Encontrará la información sobre el uso de esta Interfaz gráfica (R.O.LE.S.) en el
documento: "Manual_ROLES.doc"','R.O.LE.S.','help')
open ROLES_gui_man.htm
% --------------------------------------------------------------------
function config_user_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_user_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function config_user_derecho_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_user_derecho_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.usuario_zurdo=0;
set(hObject,'Checked','on');
set(handles.config_user_zurdo_mnu,'Checked','off');
```

```
guidata(hObject, handles);

% --------------------------------------------------------------------
function config_user_zurdo_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_user_zurdo_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.usuario_zurdo=1;
set(hObject,'Checked','on');
set(handles.config_user_derecho_mnu,'Checked','off');

guidata(hObject, handles);


% --- Executes on button press in archivo_rbt.
function archivo_rbt_Callback(hObject, eventdata, handles)
% hObject    handle to archivo_rbt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of archivo_rbt

set(handles.archivo_rbt,'Value',1)
set(handles.webcam_rbt,'Value',0)
set(handles.abrir_btn,'Enable','on')
set(handles.continuo_ckb,'Enable','off')
closepreview(handles.obj)
axes(handles.imagen_axe)
handles.himagen=imshow(handles.imagen);

if isempty(handles.imagen)
   set(handles.reconocer_btn,'Enable','off')
else
   set(handles.reconocer_btn,'Enable','on')
end

guidata(hObject, handles);

% --- Executes on button press in webcam_rbt.
function webcam_rbt_Callback(hObject, eventdata, handles)
% hObject    handle to webcam_rbt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of webcam_rbt

% global handles2

set(handles.webcam_rbt,'Value',1)
set(handles.archivo_rbt,'Value',0)
set(handles.abrir_btn,'Enable','off')
set(handles.continuo_ckb,'Enable','on')
```

```
handles.obj = videoinput('winvideo');
pos=get(handles.obj,'ROIPosition');

guidata(hObject, handles);

preview(handles.obj,handles.himagen)

set(handles.imagen_axe,'YDir','reverse')
set(handles.himagen,'YData',[1,pos(4)],'XData',[1 pos(3)])

set(handles.reconocer_btn,'Enable','on')

handles2=handles;

guidata(hObject, handles);

% --- Executes on button press in continuo_ckb.
function continuo_ckb_Callback(hObject, eventdata, handles)
% hObject    handle to continuo_ckb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of continuo_ckb


if get(handles.continuo_ckb,'Value')
   handles.t=timer('TimerFcn','reconocer_video','Period',3,'ExecutionMode','fixedRate');
   guidata(hObject, handles);
   axes(handles.bordes_axe)
   imshow(0)
   set(handles.continuo_ckb,'Value',1)
   guidata(hObject, handles);
   start(handles.t)
else
   guidata(hObject, handles);
   stop(handles.t)
end

guidata(hObject, handles);



% --------------------------------------------------------------------
function config_metodorec_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_metodorec_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function config_metodorec_ambos_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_metodorec_ambos_mnu (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.metodo_lbl,'String','Combinado');
set(hObject,'Checked','on');
set(handles.config_metodorec_rna_mnu,'Checked','off');
set(handles.config_metodorec_vp_mnu,'Checked','off');
handles.bandera_metodo = 0;

guidata(hObject, handles);

% --------------------------------------------------------------------
function config_metodorec_rna_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_metodorec_rna_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.metodo_lbl,'String','Red Neuronal');
set(hObject,'Checked','on');
set(handles.config_metodorec_ambos_mnu,'Checked','off');
set(handles.config_metodorec_vp_mnu,'Checked','off');
handles.bandera_metodo = 1;

guidata(hObject, handles);

% --------------------------------------------------------------------
function config_metodorec_vp_mnu_Callback(hObject, eventdata, handles)
% hObject    handle to config_metodorec_vp_mnu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.metodo_lbl,'String','Valores Propios');
set(hObject,'Checked','on');
set(handles.config_metodorec_rna_mnu,'Checked','off');
set(handles.config_metodorec_ambos_mnu,'Checked','off');
handles.bandera_metodo = 2;

guidata(hObject, handles);


% --- Executes on slider movement.
function handles=umbral_sld_Callback(hObject, eventdata, handles)
% hObject    handle to umbral_sld (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.umbral_txt,'String',sprintf('%0.2g',get(handles.umbral_sld,'Value')))

handles.umbral=get(handles.umbral_sld,'Value');
```

```
handles=procesar_imagen(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function umbral_sld_CreateFcn(hObject, eventdata, handles)
% hObject    handle to umbral_sld (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% % % Segmento agregado para probar refrescamiento de imagen skel en la GUI
% % % - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% % if get(handles.reconocer_btn,'Enable') == 'on'
% %     imag = handles.imagen;
% %     a_h = handles.dimensiones.a_altura;
% %     b_h = handles.dimensiones.b_altura;
% %     umb = handles.umbral;
% %
% %     [ mano, skel, p_A, p_I, p_D ] = proc_foto(imag, a_h, b_h, umb);
% %     handles.imagen_bw = skel;
% %     handles.data_mano = [p_I'; p_A'; p_D'];
% %
% %     % Presentación de imagen cargada desde archivo
% %     axes(handles.imagen_axe)
% %     %set(gcf,'NextPlot','add');
% %     handles.himagen=imshow(handles.imagen);
% %     % Presentación de imagen cargada desde archivo convertida a blanco y negro,
% %     % con recorte de áreas excedentes.
% %     axes(handles.bordes_axe)
% %     % set(gcf,'NextPlot','add');
% %     imshow(handles.imagen_bw);
% %
% %     set(handles.letra_lbl,'String','?');
% %     set(handles.reconocer_btn,'Enable','on')
% % end
% % % - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

guidata(hObject,handles)


function umbral_txt_Callback(hObject, eventdata, handles)
% hObject    handle to umbral_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of umbral_txt as text
```

```
%        str2double(get(hObject,'String')) returns contents of umbral_txt as a double

if abs(eval(get(handles.umbral_txt,'String'))) < 0.1
   set(handles.umbral_sld,'Value',eval(get(handles.umbral_txt,'String')))
else
   set(handles.umbral_sld,'Value',0)
   set(handles.umbral_txt,'Value',0)
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function umbral_txt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to umbral_txt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes when user attempts to close roles.
function roles_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to roles (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if ~isempty(handles.t)
   delete(handles.t)
end

% Hint: delete(hObject) closes the figure
delete(hObject);

function handles=procesar_imagen(hObject, eventdata, handles)

if handles.usuario_zurdo
   handles.imagen=fliplr(rgb2gray(handles.imagen));
end

imag = handles.imagen;
a_h = handles.dimensiones.a_altura;
b_h = handles.dimensiones.b_altura;
umb = handles.umbral;

[ mano, skel, p_A, p_I, p_D ] = proc_foto(imag, a_h, b_h, umb);

if isempty(mano)
   return
end
```

```
handles.imagen_bw = skel;
handles.data_mano = [p_I'; p_A'; p_D'];

% Presentación de imagen cargada desde archivo
axes(handles.imagen_axe)
%set(gcf,'NextPlot','add');
handles.himagen=imshow(handles.imagen);
% Presentación de imagen cargada desde archivo convertida a blanco y negro,
% con recorte de áreas excedentes.
axes(handles.bordes_axe)
% set(gcf,'NextPlot','add');
imshow(handles.imagen_bw);

set(handles.letra_lbl,'String','?');
set(handles.reconocer_btn,'Enable','on')

% guidata(hObject,handles)
```

12.3.13 vert.m

```
function [b] = vert( a )

b=a;
r = size(a,1);
c = size(a,2);

for i = 4: (r-4);
   for j =1 : c;
      if a(i,j)==1
         if a(i-1,j)==0 & a(i+1,j)==0 ||  a(i-2,j)==0  & a(i+2,j)==0 || a(i-3,j)==0  & a(i+3,j)==0
            b(i,j)=0;
         end
      end
   end
end
```