IDENTIFICACIÓN FACIAL PARA SISTEMAS COMPUTARIZADOS DE CONTROL DE ACCESO DE PERSONAS UTILIZANDO REDES NEURONALES

FACIAL IDENTIFICATION TO COMPUTERIZED CONTROL SYSTEMS FOR ACCESSING PEOPLE USING NEURAL NETWORKS

Mario Borja[1], Edgar Cabana Vilchez[2], Martín Montes[3], Nilton Cuellar [4], Drago A. Separovich Murata[5], Javier A. Rojas Tintaya[6], Rudolph H. Molero[7]

RESUMEN

En el presente proyecto se aplicará reconocimiento de imágenes mediante redes neuronales para la identificación facial, lo cual será utilizado para la implementación de sistemas computarizados de control de acceso de personas, donde se realizará la identificación de rostros de personas, que será captado en tiempo real con cámaras ubicadas en puntos estratégicos de acceso en áreas restringidas. En base a esta experiencia se planteará la solución al problema de identificación facial, basado en el problema de reconocimiento de patrones. Cabe mencionar que la presente investigación, además de resolver el problema de identificación facial también se plantea crear un software en forma de librerías o módulos que pueden ser usados para futuras aplicaciones en lenguaje de uso común como el C++ en cualquiera de sus presentaciones y con esto se espera crear una base para el desarrollo de software comercial. Y por último también pretendemos analizar el problema general de reconocimiento de patrones, puesto que se podría hacer un reconocimiento a muchos productos, utilizando el mismo principio, que usamos en este proyecto.

Palabras clave.- *Back-propagation, Red neurona artificial, Procesamiento de imagen.*

ABSTRACT

The current project centers its attention on image recognition by using neural networks. It will be used to implement an access control system for securing areas. Basically, it recognizes people faces in real time with cameras situated on the access entrances to restricted areas. It is important to remark that the current research, aside from solving the problem of facial recognition, proposes the development of software in libraries or modules format – written in C++ language or other common languages, which could be used in further applications. It is the intention of this initiative to create a base for commercial software development and for last we like to analyze the general problem of patron recognition, because we can do recognition a lot of products and could use the equal principle, which we used in this project.

Key words.- *Back-propagation, Artificial neural network, Image processing.*

[1]M.Sc. Docente investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[2]Ing. Docente investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[3]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[4]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[5]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[6]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería,
[7]Alumno investigador de la Facultad de Ingeniería Mecánica de la Universidad Nacional de Ingeniería.

INTRODUCCIÓN

Este proyecto surgió ante la necesidad de dar una solución a la seguridad ciudadana y también a diferentes organizaciones que lo requieran. El reconocimiento de patrones a partir de un conjunto de características (visuales o de otro tipo) puede ser una tarea complicada, que requiere la utilización de modernas técnicas de la denominada Inteligencia Artificial. Una de las técnicas más prometedoras son las Redes Neuronales Artificiales, que emulan el modo de procesar de nuestro cerebro, en el que pasamos inicialmente por una fase de aprendizaje del mundo exterior y posteriormente utilizamos esta información para tomar decisiones en el reconocimiento de nuevos patrones que se nos presentan.

El presente proyecto permitirá iniciar el desarrollo de tecnología en nuestro medio de última generación en el área de identificación biométrica, utilizando un medio de captación de características biométricas de bajo costo como es una cámara, lo cual permitirá reducir el costo para la implementación de sistemas de identificación de personas.

Por otro lado, este trabajo permitirá integrarnos a los problemas que requieran de procesamiento de imágenes, los cuales darían origen a la implementación de futuros proyectos, tales como: la selección de cualquier tipo de productos agrícolas dada esta nueva tendencia de exportación en el país, pues, aquí planteamos una nueva alternativa de solución para la selección de los mismos, y por último se encontraría también una aplicación en la geología, para identificar zonas de alto riesgo.

OBJETIVOS

- Desarrollar un prototipo de software computacional que permita realizar identificación facial de personas.
- Desarrollar las fases de procesamiento de imágenes con las fotos capturadas.
- Diseñar e implementar la red neuronal artificial con sus algoritmos de capacitación e identificación correspondientes.
- Crear una experiencia base para futuros proyectos en el ámbito de inteligencia artificial y tratamiento digital de imágenes.

FUNDAMENTO TEORICO PREVIO

La Red neuronal artificial

Primero Introduciremos un modelo sencillo de la neurona para luego construir redes, nuestro fin último es modelar correctamente la conducta global de toda una gran red, pues no se pretende modelar exactamente el comportamiento fisiológico de la neurona, sino más bien sólo sus características más relevantes, que entran en juego en su interacción con toda la red.

En la Fig. 1, tenemos un esquema de neurona. Nuestra neurona de interés es la yj. Las n neuronas xi están enviando señales de entradas, que son los valores numéricos de "algo". Los valores wji representan los pesos sinápticos en las dendritas de yj. Obsérvese la notación: el primer índice denota a la neurona hacia donde se dirige la información, el segundo índice denota de qué neurona procede la información [1, 2 y 5].

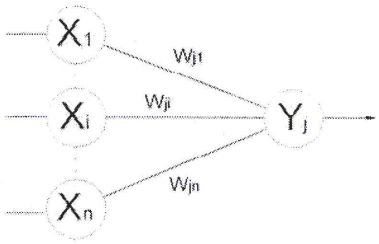

***Fig. 1** Red Neuronal artificial.*

Arquitectura de la red neuronal

Esta es la verdadera piedra angular de nuestros problemas, y en realidad la verdadera causa del inaceptable estatismo del sistema. Para muchos es ya obvio que la arquitectura capas-neuronas de una red no es nunca constante y en ocasiones ni siquiera depende del problema, pues por lo general es dada por la experiencia del entrenador [5]. Pero hay una clara idea sobre ello, que va a depender de la cantidad de personas que identificaríamos, además sabemos que el proceso de fijación de la arquitectura es un proceso de ensayo y error a veces tedioso y demorado. Si ya tenemos la idea de lo dicho, debe ser claro que este problema merece

una solución de dimensiones grandes como lo es él, pero es también claro que este problema solo surge si la que utilizamos es una red multicapa de algoritmo Back-Propagation ya que como veremos más adelante otras redes no presentarán de manera tan sesgada este problema de arquitectura. También se implementó la solución del mismo usando el Perceptrón simple, para comparar el tiempo de entrenamiento, pues la limitación computacional es determinante a la hora de implementarlas ya que al procesar estas imágenes requieren de un gran recurso computacional, y es por ello que se trata de optimizar los algoritmos al máximo.

Perceptrón simple

El Perceptrón simple se basa en una neurona que va ajustando sus pesos según las ecuaciones.

$$W^{K+1} = W^{K} + \Delta W^{K} \quad (1)$$

$$\Delta W^{K} = \alpha (t - y) p \quad (2)$$

Y la salida del Perceptrón será:

$$y = \varphi \left(\sum_{j=1}^{36000} W_j X_j \right) \quad (3)$$

En la Fig. 2, vemos la representación de la forma de trabajo de un perceptrón en cuanto se refiere a la obtención de valor de la salida, y de alguna manera nos especifica también como realizan las operaciones en un perceptrón especifico.

***Fig. 2** Red neuronal perceptrón simple.*

Red neuronal "Backpropagation"

La red neuronal back-propagation nos permite analizar con un mayor detalle, puesto que el diseño es impreciso y no existe quizá una teoría en la que se sustente el número de neuronas a elegir en la capa oculta. En este caso ha sido suficiente la implementación con solo una capa oculta.

Hablando acerca del número de neuronas a la entrada son treinta y seis mil (36000), porque es la misma cantidad de píxeles que poseen la fotos que van a ser entrenadas, luego el número de neuronas a la salida dependerá de la cantidad de personas a reconocer en este caso tomamos seis personas como referencia, para lo cual tres neuronas son suficientes, ya que con ello obtenemos ocho combinaciones a la salida.

En la Fig. 3, se muestra la arquitectura genérica para el caso del back-propagation. Pese a lo mencionado anteriormente la red back-propagation tiene algo más que es la exactitud de trabajar a punto flotante, es decir, ello nos permitirá trabajar con datos de fotografías a colores y no binarizadas.

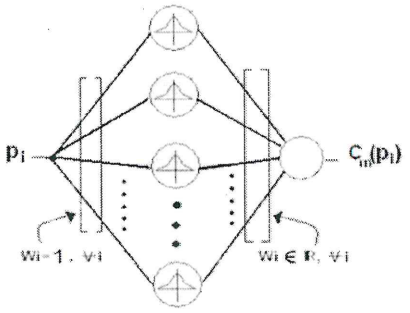

***Fig. 3** Red neuronal Back-Propagation.*

Al igual que el Perceptrón simple (en cuanto la determinación de sus parámetros), vemos que los pesos se actualizan definidos por la ecuación N° 4.

$$W^{K+1} = W^{K} + \alpha (-\nabla^{K}) \quad (4)$$

Aquí se calcula el error en cada etapa del procesamiento, lo cual nos será útil a la hora de medir la evolución del error y será un indicador que nos ayudará a ir por un buen camino.

$$(e^{K})^2 = \left\{ d^{K} - W^{K})^{T} . X^{K} \right\}^2 \quad (5)$$

El gradiente se calcula según la fórmula que sigue:

$$\nabla^{K} = \frac{\partial (e^{K})^2}{\partial W_{K}} \quad (6)$$

Aquí se muestra el gradiente del error, el cual se tomará como referencia para la evolución de la red neuronal, ya que este será determinante a la hora de actualizar los pesos donde está contenida la información del aprendizaje, lo cual se calcula según ecuación Nº 08.

$$\nabla^K = \left[\frac{\partial (e^K)^2}{\partial W_0^K} + \frac{\partial (e^K)^2}{\partial W_1^K} + \ldots + \frac{\partial (e^K)^2}{\partial W_n^K} \right] \quad (7)$$

$$W^{K+1} = W^K - \alpha \frac{\partial (e^K)^2}{\partial W_K} \quad (8)$$

Luego vemos que el error es calculado de la forma que sigue, según la ecuación Nº 09.

$$W^{K+1} = W^K - \alpha \frac{\partial (d^K - (W^K)^T X^K)^2}{\partial W_K} \quad (9)$$

Finalmente vemos en forma simplificada la actualización de los pesos.

$$W^{K+1} = W^K + 2.\alpha.e^K.X^K \quad (10)$$

IMPLEMENTACIÓN DEL SOFTWARE DE RECONOCIMIENTO FACIAL

Definición del tamaño de imágenes y contenido de datos

Inicialmente se optó por imágenes de 240x240 píxeles (lo cual significan 57600 entradas) la red entrenada y probada obteniéndose resultados satisfactorios, posteriormente se modificó el tamaño de las imágenes a 200*180 píxeles (36000 entradas), y las pruebas demostraron que los resultados seguían siendo los esperados, por lo tanto el tamaño final seleccionado fue de 200*180 píxeles.

En la Fig. 4, se aprecia la diferencia de los tamaños de las imágenes utilizadas al inicio y al final, esta reducción se hizo porque al compararlas no encontramos diferencias importantes.

Imagen 240x240 píxeles	Imagen 200x180 píxeles

Fig. 4 Reducción de imagen.

Ahora también podíamos seguir disminuyendo el tamaño pero ya la red perdía eficiencia por la resolución así que nos quedamos con estas dimensiones de foto. Para el caso de la red Perceptrón, es muy importante la selección de la función de activación en la capa oculta de la red, para ello se eligió la función Hardlims, por presentar como salidas 1 y -1, esto ayudará a que la convergencia sea más veloz a la hora del entrenamiento.

Obtención de la base de datos para el entrenamiento de la red

Se procedió a la toma de una base de datos de cinco (5) fotografías por persona, empleando una cámara de red. Este software ha sido elaborado en Visual C++, porque resulta útil su optimización en cuanto a los recursos computacionales, sin embargo existen herramientas que han sido usados como guía.

En la Fig. 5, vemos los pasos a seguir en cuanto se refiere al procesamiento de imágenes para la implementación del software. Estos son imprescindibles para adecuar las fotos óptimas que serán llevadas a la red, que se usarán para las etapas de entrenamiento y reconocimiento. Estas etapas serán explicadas posteriormente, ya que aquí solo queremos mostrar un bosquejo genérico

de las etapas de procesamiento de imágenes.

Más adelante ahondaremos qué procesos continúan luego de haber tomado la foto, aquí mencionamos las más importantes como: se extrae solo la cara, luego se amplía o se reduce según sea el caso, esto para normalización del tamaño de la foto, este paso es importante porque la red neuronal requiere de un mismo tamaño de fotos para su entrenamiento.

En la Fig. 6, se detalla gráficamente cómo se procede en una descomposición de foto a píxeles. Obteniendo así en un arreglo de información, en este caso vector, que llegarán a ser nuestros datos a la entrada de la red.

Posteriormente se llevará a un proceso de entrenamiento y finalmente la red neuronal tomará la decisión de reconocimiento.

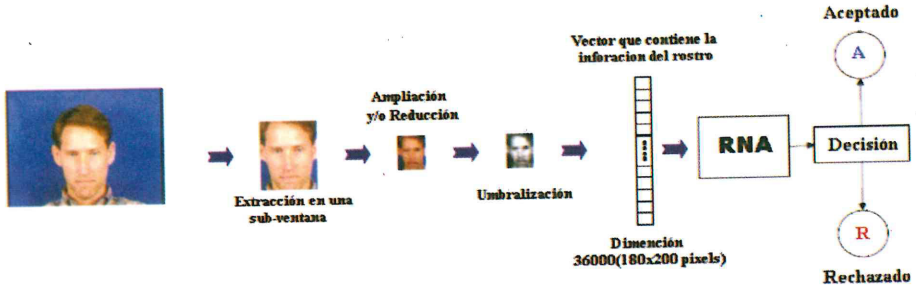

Fig. 5 Diagrama para el entrenamiento de la red.

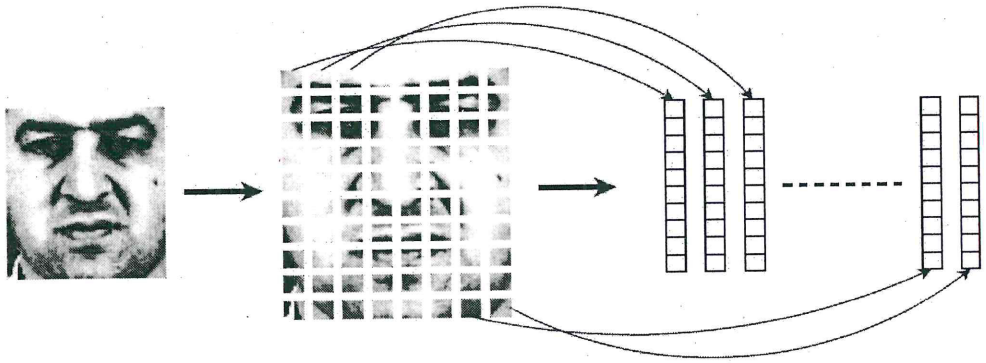

Fig. 6 Descomposición de la imagen.

Diseño del entorno y programación de la red en Visual C++

El programa ha sido desarrollado en Visual C++, usando la aplicación de MFC AppWizard, de tipo documento simple, por lo que sólo nos enfocamos hacia la demostración, aunque podría hacerse comercial, siendo así el entrenamiento un proceso previo y cambiaría de acuerdo a las personas que se van ha identificar.

El binarizado.- Es un proceso donde las fotos son convertidas al formato de blanco y negro por medio de un filtro que dependerá de las condiciones de iluminación estando controlado de un valor que será constante para todas las fotos patrones y de prueba.

La segmentación.- Es un proceso por el cual las fotos eliminan los agentes externos al rostro que no nos interesan al momento de hacer el reconocimiento.

El entrenamiento.- Es donde se procederá a cargar cada una de la fotografías de la base de datos, realizándose la segmentación y el binarizado, luego estas fotos son evaluadas por la red, así sucesivamente reduciendo el error hasta llegar a una meta.

En la Fig. 07, observamos la ventana de presentación del software desarrollado en Visual C++, consta de tres botones, dentro de las funciones vemos que el primer botón es del entrenamiento de la red neuronal, conduciendo a una ventana para poder capacitarla con los diferentes patrones, luego los datos son almacenados en archivos de formato ".dat", los cuales se almacenan en la memoria del ordenador para luego ser usados al momento de activar el segundo botón que es la de probar la red donde surgen ciertas herramientas para verificar si la red ha aprendido o no. Finalmente el botón de inicializar los pesos del perceptrón nos permite hacer un borrado total de todos los pesos entrenados del último proceso ejecutado, permitiendo verificar que no haya fraudes en la comprobación del aprendizaje de la red neuronal.

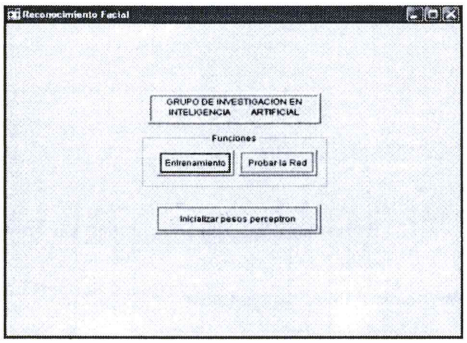

Fig. 7 *Ventana de presentación del Software.*

Como podemos apreciar es un formato muy sencillo y elemental de presentación, sin embargo no es suficiente para poder verificar paso a paso el proceso de entrenamiento y aprendizaje ya que todo el proceso complejo lo desarrolla el algoritmo codificado en lenguaje Visual C++ y nosotros nos encargamos de apreciar todas las bondades que nos proporciona. Es importante mencionar que estos procesos de reconocimiento nos ha conducido ha encontrar diferentes tipos de solución para reconocimiento general de patrones, es decir, puede extenderse hacia los medios geográficos, analizar su accesibilidad y el estudio de suelos, también en la agricultura exactamente para la selección de frutos, y por último podemos también hablar de la selección de fibra de camélidos. En la Fig. 8, mostramos la ventana de entrenamiento la cual consta de un botón "Entrenar la Red", que nos permite hacer las iteraciones por todas las fotos base que se han almacenado en memoria de la maquina, esto finalizará hasta que se consiga la meta.

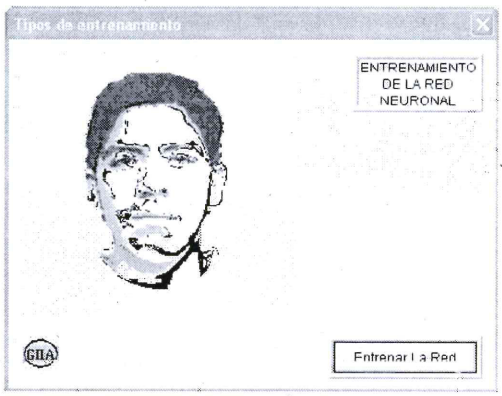

Fig. 8 *Ventana de entrenamiento para la red.*

En la Fig. 9, mostramos un fenómeno muy interesante acerca de las redes neuronales, bien sabemos que esta red de trabajo no puede representarse gráficamente por la cantidad de dimensiones que posee, sin embargo la gráfica que se muestra a continuación nos da más o menos una idea acerca de lo que hace la red neuronal. En este caso tenemos un grafo de la letra "X", vemos que se toman ciertas partes prioritarias de la foto que tenemos como información y la llevamos a la red neuronal asignándole una determinada salida, pues estamos trabajando con una red supervisada.

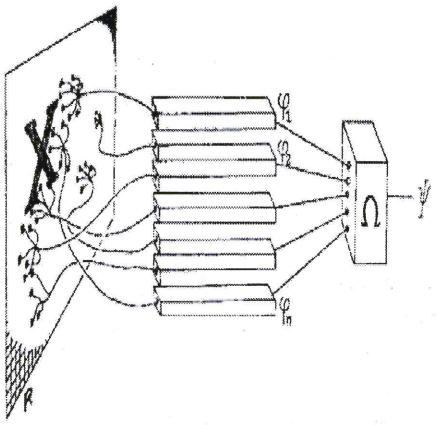

Fig. 9 *Representación del aprendizaje.*

Entonces las primas que vemos, son los pesos que tienen que contener la información del aprendizaje para finalmente dar una respuesta, un punto muy importante es que no sabemos como puede resultar este tipo de reconocimiento, ya aún está en estudio su conducta y la optimización de su arquitectura.

En la Fig. 10, apreciamos el estudio de su conducta, que se resume en la función costo "C (W)". Esto nos permite analizar cómo irá descendiendo el error y por ende un óptimo aprendizaje. En la gráfica vemos la función costo de una red cualquiera, entonces vemos que del punto G (WT), va descendiendo al punto G' (WT+1), variando los pesos para minimizar "C".

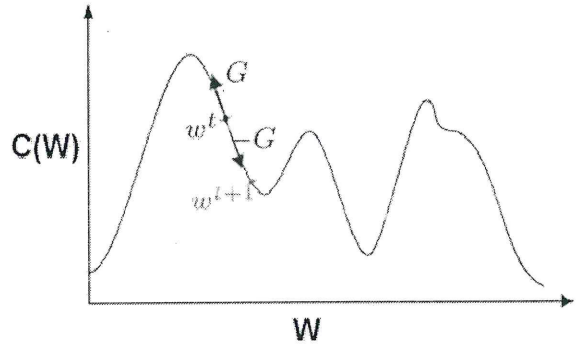

***Fig. 10** Gráfica de la función de costo.*

Como en toda curva se tiene mínimos locales y un mínimo global, sin embargo, en lo posible se trata de evitar caer en los mínimos locales, ya que los pesos generarían un falso aprendizaje, esto se evita aumentando el factor momentum o factor de olvido, lo cual resulta útil para evitar este tipo de problemas.

En la ecuación N° 11 vemos la ecuación de actualización de pesos con factor de olvido.

$$W^{K+1} = W^K + 2.\alpha.e^K.X^K + \mu\left(W^K - W^{K-1}\right) \quad (11)$$

Donde μ, factor de olvido.

Probar la red.- Esta es la función en la que verificaremos los resultados obtenidos en el entrenamiento, se cargará una nueva imagen y se espera que la persona sea identificada correctamente, de haber un error, se debe proceder a un nuevo entrenamiento.

En la Fig. 11, se muestra la ventana de prueba de la red la cual, nos permite cargar una foto diferente a las del entrenamiento para poderla evaluar y hacer el respectivo reconocimiento.

Inicialización de pesos.- Esta función sólo nos permite inicializar los valores numéricos de los pesos los cuales pueden ser aleatorios o específicos pero obtenidos de modo aleatorio.

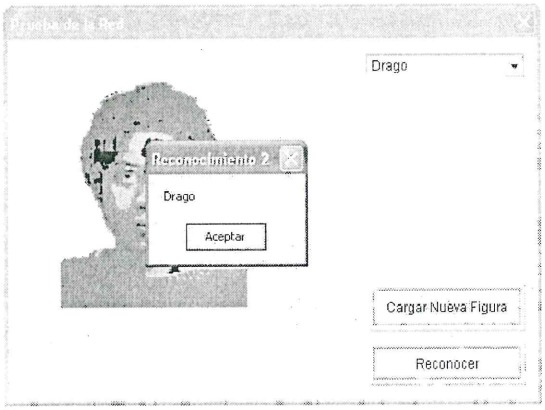

***Fig. 11** Prueba de aprendizaje.*

Procesamiento de imágenes con el Visual C++

Para poder realizar el entrenamiento de la red y para una mayor eficiencia debemos realizar un procesamiento previo de las imágenes, esto implica la segmentación, para separar el rostro del entorno y la binarización para simplificar los datos de entrada.

Los pasos en el pre-procesamiento son:

- bordeado y binarizado.

El bordeado.- Es un proceso por el cual nos permite seleccionar solo el rostro, porque es lo que nosotros queremos resaltarle más a la red neuronal, tanto para su entrenamiento como para su reconocimiento [3].

El binarizado.- Resulta importante ya que simplificará los cálculos que realizará la computadora en el proceso de aprendizaje, para esto se convertirá la imagen a color a blanco y negro, donde se elegirá un umbral, a partir del cual un píxel cuyo valor sea menos al umbral será considerado blanco o se le asignará un valor de -1, y aquellos píxeles cuyo valor sea mayor a dicho umbral será considerado negro y se le asignará el valor de 1[1].

En la Fig. 12, podemos apreciar cómo es que seleccionamos sólo el rostro con la ayuda de un diferenciador de colores, ello nos permite hacer un bordeado muy bueno luego es binarizada y luego será llevada a la entrada de la red.

La segmentación y extracción del rostro es un paso muy importante ya que esto elimina mucha información innecesaria al momento del entrenamiento mejorando la eficiencia de la red.

El algoritmo de entrenamiento back-propagation es mucho más eficiente que la red perceptrónica de una capa y converge más rápido, característica que nos hace pensar en bancos más grandes de personas.

Se comprobó que no es necesario usar la correlación cruzada para centrar la imagen debido a la plasticidad de la red, esto es muy importante ya que nos ahorra un montón de programación y además un desgaste del procesador.

REFERENCIAS

1. **Kulkarni, A. D.**, "Computer Vision and Fuzzy-Neural Systems". Prentice Hall, New Jersey, 2001.
2. **Uzarieta, F., Saavedra, C.**, "Redes Neuronales Artificiales". Departamento de Física, Universidad de Concepción.
3. **Gil, P., Torres, F. Ortiz, F. G.**, "Detección de objetos por segmentación multinivel combinada de espacios de color". Universidad de Alicante, 2004, España.
4. **Gonzáles, R. C., Woods, R. E.**, "Digital Image Processin". Prentice Hall, New Jersey, 2002.
5. **Valluru, B. R.**, "C++ Neural Networks and Fuzzy Logic". IDG Books Worldwide, 1995.

Correspondencia: mecamolfer@hotmail.com

Recepción de originales: Julio 2007
Aceptación de originales: Octubre 200