Universidad Central “Marta Abreu” de Las Villas.
Facultad Matemática, Física y Computación
Licenciatura en Ciencia de la Computación

TRABAJO DE DIPLOMA

Automatización del reconocimiento de imágenes en las fotos digitales de los metros contadores del tráfico telefónico en centrales analógicas de ETECSA.

AUTORES: Dagnier A. Curra Sosa
TUTOR(ES): Isis Bonet Cruz
María Matilde García Lorenzo
CONSULTANTE:
Jacqueline García Bermúdez

“Año del 50 Aniversario del Triunfo de la Revolución”

FECHA DE REALIZACION DE LA DEFENSA
1 de julio del 2009

Dictamen.

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del tutor

Firma del jefe del Laboratorio

A mi mamá, por hacer posible mi existencia.
A la que debo toda mi educación y mi persona.
Por su incondicional amor y apoyo.
A mi familia, por toda la confianza depositada.
Por su enorme paciencia en estos 5 años de estadía lejos de casa.
A todas aquellas amistades cuyo aporte ha contribuido a mi bienestar.

A mis tutoras Marilín e Isis,

Por todo el esfuerzo, la dedicación, el derroche de empeño y voluntad del que han hecho gala; por sus consejos y ánimos en los momentos difíciles.

A Leticia y Morell por su preocupación y seguimiento de mi trabajo.

A Enrique Casanovas por su amistad e incondicional ayuda en la implementación y validación de los resultados.

A la Filial de Tecnología y Software de ETECSA de Holguín por su asesoría, transportación y contribución de material de pruebas.

A mis hermanos de Holguín por su preocupación y aporte significativo.

A todas mis amistades en general de la UCLV, mis compañeros de clases y facultad, mis profesores y mis amigos fieles del vicio las cuales tienen su lugar en mi corazón;

Por los momentos agradables, su contribución a mi formación, sus consejos, sus criticas constructivas y sin lugar a dudas su maravillosa amistad.

A todos aquellos que han contribuido de alguna forma a la realización de esta tesis.

Resumen

Este trabajo surge a partir de la problemática planteada por la Gerencia de ETECSA de la provincia de Holguín y tiene como objetivo humanizar el trabajo de facturación de tarifas para clientes de teléfonos analógicos. Se parte de las fotografías de más de 100 pizarras de metrocontadores existentes en esta provincia, donde todas las fotos están almacenadas en formato JPG y en las cuales diversos efectos de ruido conspiran en una captación adecuada de los dígitos que conforman cada lectura. A partir de estas se diseñan e implementan las distintas fases concernientes a elaborar un OCR.

La imagen es llevada a un formato binario y se implementa como método de segmentación el Método Basado en Píxeles. La compresión desarrollada es la codificación Huffman y RLE, atendiendo a sus bondades con los formatos JPEG.

Como reconocedor se implementa un modelo de Holpfield Discreto que incluye cuatro variantes de calcular los pesos y dos modos de funcionamiento: secuencial asíncrono y paralelo síncrono. Este reconocedor fue validado usando un método hold-out.

Finalmente se concibe una interfaz para el OCR sencilla y asequible y que permite dar respuesta a los requerimientos del usuario final.

Abstract

This work emerges from the problems raised by the Management ETECSA of Holguin province and aims to humanize the work of billing rates for customers of analog phones. It is part of the photographs of more than 100 metrocontadores boards existing in this province, where all photos are stored in JPG format and in which various effects of noise conspire in an appropriate collection of digits that make up each reading. Since these are designed and implemented various stages relating to developing an OCR.

The image is brought into a binary format and is implemented as a method of segmentation method based on pixels. The developed compression is Huffman coding and RLE, according to their strengths with JPEG.

Recognizer is implemented as a model that includes four Holpfield Discrete variants of calculating the weights and two modes: asynchronous and parallel synchronous sequential. This recognition was validated using a hold-out method.

Finally is an interface designed for simple, affordable and OCR to respond to the requirements of the end user.

Índice

Introducción

En todas las direcciones territoriales del país pertenecientes a la Empresa de Telecomunicaciones y Electrónica de Cuba (ETECSA) existen Centrales Analógicas que poseen metros contadores por cada servicio telefónico para medir la utilización del tráfico telefónico de larga distancia nacional. Mensualmente se le tiran fotos con cámara digital a los mismos, estas reflejan la lectura de cada uno de ellos, y se usan para el cálculo del importe a facturar de cada servicio por este concepto. Existe un software que permite la captación por visualización directa en la pantalla y permite ampliar las fotos y/o focalizar a la vez la lectura de un metro en específico. Esta captación la hace una operadora de forma visual, lo cual puede introducir errores, ya sea al observar o al teclear el número, así como demorar el procesamiento de los importes, debido a la cantidad de imágenes a procesar.

La solución a esta problemática es la confección de un sistema de Reconocimiento Óptico de Caracteres (OCR), que permita reconocer la salida de los metros contadores de forma automática, a partir de las imágenes que se le toman a los mismos, para almacenarlos en un formato con el cual puedan interactuar programas de edición de texto.

Sobre este campo en desarrollo se han hecho numerosos aportes entre los cuales podemos citar a algunas de las diferentes opciones que existen en el mercado: Omnipage Pro, WorkingPapers y IRISPen; algunas más actuales como Abbyy FineReader 9.0 Professional Edition, OneNote 2007, TopOCR 1.0 y Omniformat 8.1; sin dejar de mencionar aquellas que son libres tales como Kooka, GOCR, Ocrad, Ocre, Tesseract y OCRopus.

Es válido señalar que la incorporación de estos métodos en aplicaciones resulta, económicamente, bastante costosa. A pesar de esta dificultad también tiene sus ventajas en cuanto al número de documentos que pueden procesar y la calidad que brindan en un largo plazo de tiempo pudiendo amortizar sus gastos rápidamente.

La opción de usar un reconocedor de caracteres es preferible a la trascripción manual que, aunque es mucho más barata, requiere de mucho tiempo y energía de un especialista el cual mediante la utilización del teclado está expuesto a cometer errores, además de que debe leer una segunda vez el documento para la rectificación de su labor. De ahí que la automatización de esta tarea sea imprescindible y la implementación de un OCR se considera la solución a ser utilizada en las oficinas del Departamento de Tecnologías de Información y Software (TISW) de ETECSA. Por esta razón se plantea como objetivo general:

- Desarrollar un OCR que permita automatizar el proceso de lectura de los metros contadores para el cálculo del importe telefónico a facturar por cliente, a partir de fotos digitales de las centrales analógicas de ETECSA, mediante el uso de técnicas de procesamiento de imágenes y redes neuronales.

Objetivos específicos:

- Segmentar la imagen en fragmentos que contengan los elementos de interés y separarlos entre si para la obtención de los caracteres a identificar y definir el método compresión a emplear.
- Diseñar y entrenar una red neuronal para reconocer los caracteres que contienen las imágenes.
- Implementar un OCR para la lectura de los metros contadores.
- Validar los resultados que se alcanzan.

De acuerdo a estos objetivos se consideraron como tareas de investigación las siguientes:

- Seleccionar el método de segmentación más apropiado según las características de la imagen.
- Descomprimir la información guardada por el formato JPEG de las imágenes a tratar mediante la aplicación inversa de su algoritmo de compresión.
- Decodificar el código descomprimido a través de las técnicas codificadoras que se utilizan en la compresión JPEG.

- Obtener los valores de los rasgos descriptores en la matriz binaria de pixeles que serán usados como entrada a la herramienta de clasificación.
- Describir y seleccionar el clasificador a utilizar que mejor procese los parámetros descriptores considerados.
- Comprobar la eficiencia de la aplicación construida con el modelo de clasificador escogido.

Las hipótesis de investigación que fueron fijadas atendiendo a todo lo anteriormente expuesto, y enmarcándonos dentro de la resolución de problemas de reconocimiento de caracteres, son las siguientes:

1. La matriz binaria de píxeles nos brinda una descripción de los caracteres a un alto nivel que permite establecer diferencias entre ellos a través de la distribución de blancos y negros.
2. La red de Hopfield es un clasificador adecuado para el reconociendo de caracteres a partir de imágenes de metros contadores de ETECSA.

Este Trabajo de Diploma esta dividido en 3 capítulos, los cuales resumen y ejemplifican las técnicas y algoritmos utilizados para la resolución y cumplimiento de los objetivos antes planteados.

- **Capítulo 1:** se describe los procesos esenciales para un reconocimiento de caracteres, se hace énfasis en el procesamiento de la imagen digital como primer paso para el funcionamiento de un OCR. Los aspectos relativos a procesos de segmentación, análisis de los métodos de compresión como descifradores de la información y técnicas de clasificación a emplear son analizados en el presente capítulo.
- **Capítulo 2:** se diseña e implementa un OCR a partir de las fotos tomadas a los metro-contadores de la Gerencia de ETECSA en Holguín y tomando como reconocedor un modelo de red neuronal Hopfield Discreto

- **Capítulo 3:** En este capítulo se valida el clasificador implementado y se explican las bondades del software. Finalmente se muestra el manual de usuario.

Capítulo I: "Aspectos sobre procesamiento de imágenes y su reconocimiento".

En este capítulo se describen los conceptos y métodos fundamentales para el procesamiento de una imagen digital. Los aspectos relativos a procesos de segmentación, y compresión como descifradores de la información se abordan profundizando en los de más interés. Finalmente se tratan varias técnicas de clasificación factibles a utilizar.

1.1 Reconocimiento Óptico de Caracteres.

La visión computacional o artificial es uno de los grandes campos de la aplicación de las técnicas de procesamiento y análisis de imágenes, las cuales, necesariamente para hacer esta tarea, tienen que ser combinadas con técnicas de otras ramas del saber, como la IA, Reconocimiento de Patrones, Estadística, etc.

El Reconocimiento de Patrones es una parte esencial de cualquier sistema de análisis de imágenes de alto nivel. La meta de un sistema de visión computacional es analizar imágenes en una escena dada y reconocer el contenido de éstas.

Un sistema de visión computacional puede definirse como un sistema, que luego de cierto proceso de "aprendizaje visual", logra reconocer los diferentes objetos o elementos de una escena nueva, es decir, realiza una interpretación automática de ésta. Dentro de dichos sistemas podemos ubicar a los de Reconocimiento Óptico de Caracteres (OCR, Optical Character Recognition).

Un OCR es una herramienta que se desarrolla para reconocer un conjunto de caracteres con determinados estilos. El método inicia con una imagen que contiene algún texto, y a través de un pre-procesamiento de la imagen, la información de interés contenida en ella es traducida a un formato de texto. La gran variedad de tipos de letras que existen para escribir a máquina hace que sea más difícil esta tarea. En los sistemas de reconocimiento de caracteres existen tres procesos esenciales:

- **Proceso de Representación**: donde la imagen es tratada por diferentes vías para lograr tener los datos en una forma de alto de nivel. La imagen debe ser

fragmentada para separar sus elementos, siendo obtenido un grupo de pequeñas imágenes como resultado, cada una de las cuales puede representar un carácter. La extracción de características especiales y patrones de la imagen ponen los datos digitales a un nivel mayor. Este proceso se denomina extracción de rasgos.

- **Proceso de Generación de una Base de Conocimientos**: contiene una representación de alto nivel de todos los caracteres tradicionales. Este es el paso donde el sistema aprende como reconocer las clases.
- **Proceso de Identificación–Clasificación**, dada la representación de alto nivel y la información aprendida de la base de conocimientos, se clasifica el carácter desconocido. Para ello se cuenta con diferentes métodos de identificación(Bonet, 2001).

El objetivo de un OCR es tomar una imagen que representa un texto (por ejemplo una imagen escaneada) y procesarla hasta llegar a convertirla en un fichero texto. El trabajo parte de la lectura de ficheros de imágenes e incluye el procesamiento para la búsqueda y tratamiento de componentes, hasta llegar a la estructura mínima que representa a un carácter, luego se usan técnicas de IA para reconocer la imagen como la letra o símbolo que representa.

Para que el sistema computacional pueda realizar el aprendizaje, se necesita la especificación de un conjunto de rasgos o descriptores, en función de los que serán caracterizados cada uno de los objetos del universo en estudio. En el caso de un OCR estos objetos son los caracteres.

El buen desempeño de un sistema de visión computacional depende en gran parte de las componentes que lo forman, existiendo 5 partes primordiales para que el sistema funcione adecuadamente (Figura 1.1).

- **Captación:** Es el proceso a través del cual se obtiene una imagen visual.
- **Preprocesamiento:** Incluye técnicas tales como la reducción de ruido y realce de detalles.

- **Segmentación**: Es el proceso que divide a una imagen en objetos que sean de nuestro interés.
- **Descripción**: Es el proceso mediante el cual se obtienen características convenientes para diferenciar las clases de objetos.
- **Reconocimiento**: Es el proceso que asocia un significado a un conjunto de objetos reconocidos (Ysiquio, 2004).

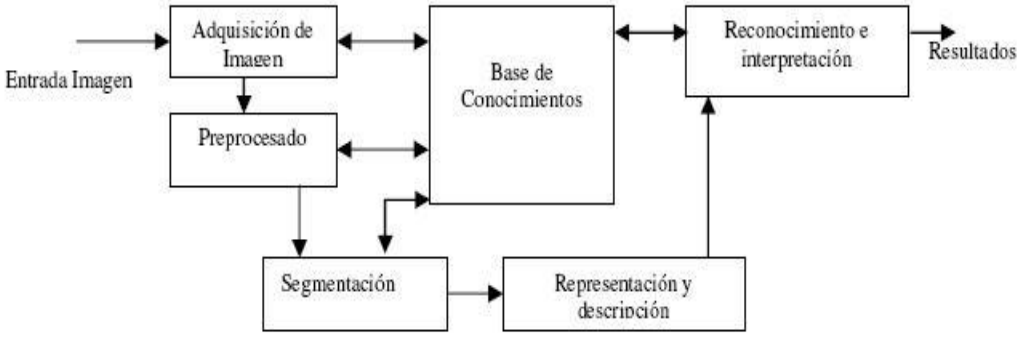


Figura 1.1 Componentes de un sistema de visión computacional.

Mientras que en una imagen los caracteres se describen indicando cada uno de los puntos que los forman, al convertirlos a un formato de texto (por ejemplo ASCII o Unicode), pasan a describirse por un solo número, por lo que se produce una reducción significativa del espacio en memoria que ocupan.

En definitiva para completar la funcionalidad del OCR un primer paso es la lectura de los ficheros y el ir analizándolos de acuerdo a la descripción del formato de imagen que los representan, para llevarlos a una representación común en forma de matriz binaria, en la cual cada elemento representa un píxel. Entonces corresponde la detección y extracción de las zonas de interés y luego de los caracteres dentro de la misma. En un segundo paso, comenzando desde la sub-matriz que describe el posible carácter una serie de parámetros de forma, distribución de blancos y negros, posición dentro de la línea, etc. pueden ser calculados en dependencia del problema, permitiendo llegar a una detallada descripción de las porciones de las imágenes. En un tercer paso esto constituye la entrada a una

herramienta de clasificación la cual es la que finalmente realiza el reconocimiento y devuelve el código del elemento más cercano a la descripción recibida de los rasgos de la imagen mínima. Una vez que estos códigos son reconocidos y almacenados, son llevados al formato texto de salida(Bonet et al., 2002).

1.2 Definición de imagen digital.

Se entiende por imagen, una secuencia en dos dimensiones

$$x\,[n_1, n_2],\ 0< n_1<N_1,\ 0< n_2<N_2 \qquad (1.1)$$

con extensión finita N_1 y N_2 en la dirección vertical y horizontal, respectivamente. Como se muestra en la Figura 1.2, los índices n_1 y n_2 indican la fila y la columna que ocupa una muestra (*píxel*) dentro de la imagen.

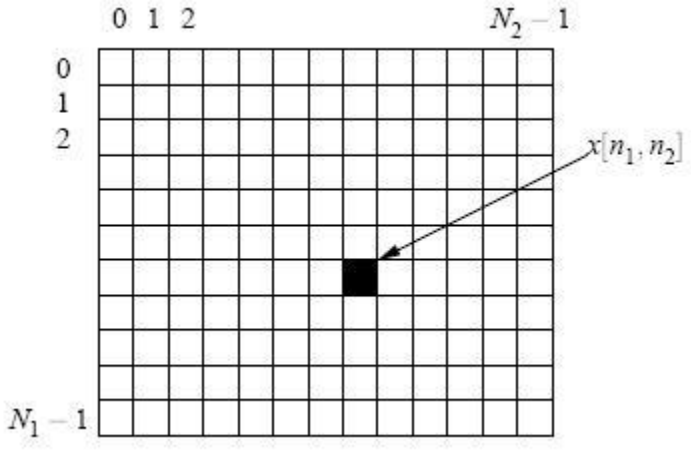


Figura 1.2 Muestras de una imagen.

En las imágenes en niveles de grises cada valor x $[n_1, n_2]$ representa la intensidad (brillo) de la imagen como promedio resultante de una función de varias variables, entre las que se incluyen profundidad, tiempo y color; redondeado al entero más cercano. Los valores de las muestras son enteros con signo

$$[\mathrm{n1}, \mathrm{n2}] \in \{-2^{B-1}, -2^{B-1}+1, \ldots, 2^{B-1}-1\}) \qquad (1.2)$$

o sin signo

$$(x[n1, n2] \in \{0,1, \dots, 2^B - 1\}) \quad (1.3)$$

con precisión de B bits. Lo más común son la imágenes sin signo con B = 8 bits. Cuando B=1 se trata de una imagen binaria, en este caso existen sólo dos niveles de gris (blanco y negro) y se representan con 0 ó 1. Las imágenes en color se representan normalmente con tres valores por cada muestra, correspondiéndose cada valor con uno de los tres colores primarios: rojo, verde y azul (en inglés *Red*, *Green* y *Blue*, *RGB*). Representamos estas imágenes con tres secuencias separadas $x_R [n_1, n_2]$, $x_G [n_1, n_2]$ y $x_B [n_1, n_2]$ aunque en general podemos tener C componentes

$$x_c [n_1, n_2], c=1,2,\dots,C. \quad (1.4)$$

Por ejemplo, las imágenes preparadas para imprimir suelen tener cuatro colores: cián, magenta, amarillo y negro; de hecho algunas impresoras de altas prestaciones añaden el verde y el violeta (Ysiquio, 2004).

Según el contenido distinguimos los siguientes tipos de imágenes:

- **Naturales**: son imágenes que representan escenas naturales, incluyendo fotografías.
- **Texto**: imágenes que representan textos escaneados o generados por computadoras, por ejemplo faxes.
- **Gráficos**: imágenes escaneadas o generados por computador al estilo de los cómic.
- **Compuestas**: imágenes que contienen una mezcla de los tipos anteriores (Vázquez, 2004).

1.3 Procesos para la segmentación de imágenes.

"El propósito de la segmentación es particionar una imagen en regiones significativas, la definición de región significativa está en función del problema considerado" (Russ, 1991).

La mayor parte de los algoritmos de reconocimiento están escritos a partir de imágenes binarias, por lo que se hace conveniente el paso de una imagen en niveles de gris (o color) a una binaria, además esto permite reducir el volumen de los datos a tratar.

La binarización de una imagen digital consiste en convertirla a una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen.

Uno de los métodos para poder binarizar una imagen digital es mediante el histograma de dicha imagen. A través del histograma obtenemos una gráfica donde se muestra el número de píxeles por cada nivel de gris que aparecen en la imagen. Para binarizar la imagen, se deberá elegir un valor adecuado dentro de los niveles de grises (umbral), de tal forma que el histograma forme un valle en ese nivel. Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los mayores en blanco.

Una vez obtenida la imagen binaria se deberá fragmentar o segmentar en las diferentes componentes conexas (parte de la imagen donde todos los píxeles son adyacentes entre sí) que la componen.

La fragmentación o segmentación de la imagen constituye una de las mayores dificultades del reconocimiento, y se hace necesaria para poder reconocer cada uno de los caracteres de la imagen binaria. Se define como la operación que permite la descomposición de un texto en diferentes entidades lógicas, las cuales deben ser lo suficientemente invariables, para ser independientes de la fuente, y lo suficientemente significativas para su reconocimiento. Con la segmentación se localizan las zonas de interés y se separaran dichas zonas. Estas estarán caracterizadas por atributos comunes como son dimensión, superficie, densidad, inclinación, textura, etc.

El objetivo de la segmentación de imágenes, es el de particionar las mismas en algunas zonas de interés correspondientes a objetos presentes en la escena. El proceso de segmentación se encarga de evaluar si cada píxel de la imagen pertenece o no al objeto de interés. Esta técnica de procesamiento de imágenes genera una imagen binaria, donde los píxeles que pertenecen al objeto se representan con un uno, mientras que los que no

pertenecen al mismo se representan con un cero. Este tipo de particionamiento está basado en el análisis de alguna característica de la imagen, tal como los niveles de gris o la textura.

Según el grado de asociación entre las operaciones de segmentación y las de reconocimiento, se distinguen tres tipos principales de métodos de segmentación:

- **Los métodos explícitos o segmentación en unidades físicas:** Estos métodos, intervienen avanzando el proceso de reconocimiento. Las partes segmentadas se dividen prácticamente en letras, tanto que la segmentación se considera una parte del proceso de reconocimiento.
- **Los métodos de segmentación implícitos o segmentación en unidades lógicas:** consisten generalmente en una segmentación más fina y así conseguir los puntos de corte correctos. Las partes segmentadas son llamadas grafemas. Estos se usarán más adelante, durante el proceso de reconocimiento/clasificación. Los grafemas estarán compuestos por fragmentos de caracteres, caracteres o grupos de caracteres.
- **Los métodos de segmentación implícitos y exhaustivos:** En este caso, es el reconocimiento quien guía la segmentación, así que el sistema de evaluación que se aplica aquí implica un reconocimiento por cálculo de las posiciones sucesivas de la imagen y escoger las posiciones de segmentación que se correspondan con las responsables de las partes más significativas.

Estos algoritmos de segmentación de imágenes se basan en alguna de las tres propiedades siguientes:

- **Discontinuidad** en los tonos de gris de los píxeles de un entorno, que permite detectar puntos aislados, líneas y aristas (bordes).
- **Similaridad** en los tonos de gris de los píxeles de un entorno, que permite construir regiones por división y fusión, por crecimiento o por umbralización.

- **Conectividad** de los píxeles desempeña un papel importante en la segmentación de imágenes. Recordemos que una región D se dice *conexa* o conectada si para cada par de píxeles de la región existe un camino formado por píxeles de D que los conecta. Un *camino* de píxeles es una secuencia de *píxeles adyacentes* (que pertenecen a su entorno inmediato).

Los métodos de segmentación más conocidos son:

Método Basado en Píxeles: Esta operación consiste en definir un rango de niveles de brillo en la imagen original, considerar todos los puntos en este rango pertenecientes a los objetos y el resto al fondo. Este método de segmentación toma en cuenta sólo el valor de gris de un píxel, para decidir si el mismo pertenece o no al objeto de interés. Para ello, se debe encontrar el rango de valores de gris que caracterizan dicho objeto, lo que requiere entonces establecer los umbrales interactivamente observando la imagen o a través de la búsqueda y el análisis del histograma de la imagen.

Método Basado en Contornos: El método basado en contornos puede ser usado para evitar la variación del tamaño del objeto. Este método se basa en realizar la búsqueda del valor máximo del gradiente, sobre cada línea que forma la imagen. Cuando un máximo es encontrado, un algoritmo de trazado trata de seguir el máximo del gradiente alrededor del objeto, hasta encontrar de nuevo el punto inicial, para luego buscar el próximo máximo en el gradiente(Bonet, 2001).

Método Basado en Regiones: Los métodos de segmentación basados en regiones, toman en cuenta un conjunto de puntos de la imagen, a los cuales se les analiza características como, la posición en el espacio de intensidades, las relaciones topológicas (conectividad) y las características de las fronteras entre dos conjuntos. Dependiendo de como sea analizada la posición en el espacio y las relaciones espaciales existentes entre los píxeles, se pueden encontrar métodos de Clasificación y métodos por Crecimiento de Regiones.

1.4 Compresión de datos.

La compresión de datos se define como el proceso de reducir la cantidad de datos necesarios para representar eficazmente una información, es decir, la eliminación de datos redundantes. En el caso de las imágenes, existen tres maneras de reducir el número de datos redundantes: eliminar código redundante, eliminar píxeles redundantes y eliminar redundancia visual.

Redundancia de Datos: Hay que resaltar la diferencia entre *información* y *datos*, ya que en muchas ocasiones se utilizan como sinónimos y no lo son. Los datos son una forma representar la información; así, una misma información puede ser representada por distintas cantidades de datos. Por tanto, algunas representaciones de la misma información contienen datos redundantes.

Código Redundante: El código de una imagen representa el cuerpo de la información mediante un conjunto de símbolos. La eliminación del código redundante consiste en utilizar el menor número de símbolos para representar la información. Las técnicas de compresión por codificación de Huffman y codificación aritmética utilizan cálculos estadísticos para lograr eliminar este tipo de redundancia y reducir la ocupación original de los datos.

Píxeles Redundantes: La mayoría de las imágenes presentan semejanzas o correlaciones entre sus píxeles. Estas correlaciones se deben a la existencia de estructuras similares en las imágenes, puesto que no son completamente aleatorias. De esta manera, el valor de un píxel puede emplearse para predecir el de sus vecinos.

Redundancia Visual: El ojo humano responde con diferente sensibilidad a la información visual que recibe. La información a la que es menos sensible se puede descartar sin afectar a la percepción de la imagen. Se suprime así lo que se conoce como redundancia visual. La eliminación de la redundancia esta relacionada con la cuantificación de la información, lo que conlleva una pérdida de información irreversible. Técnicas de compresión como JPEG, EZW o SPIHT hacen uso de la cuantificación.

La compresión de datos es beneficiosa en el sentido de que el proceso de compresión-transmisión-descompresión es más rápido que el proceso de transmisión sin compresión. Podemos expresar la ganancia en velocidad con una fórmula sencilla:

$$\mu = \frac{2*\frac{D}{c}+r*\frac{D}{b}}{\frac{D}{b}} = \frac{2*b+r*c}{c} \qquad (1.5)$$

donde μ es la relación entre el tiempo que se tardaría para transmitir comprimiendo y sin comprimir, *c* es la velocidad de compresión en bits por segundo (bps), *D* es el número de bits que componen el mensaje a transmitir, *b* es la velocidad en la línea en bps y *r* es el radio medio de compresión del algoritmo utilizado, que se puede escribir como bits comprimidos / bits totales.

Vemos que en el último término la fórmula no depende de la cantidad de datos a transferir *D*, sino de las distintas velocidades de compresión y emisión. Siempre que μ sea menor que 1, se conseguirá ventaja en velocidad al comprimir la información. Si suponemos que μ = 0.5 el tiempo de transmisión utilizando la compresión es la mitad que sin usarla.

La compresión es igualmente útil no sólo en la transmisión sino también en el almacenamiento masivo de información donde esta necesidad crece por encima del crecimiento de los discos duros o memorias que alcanzan unidades de medidas superiores a los Terabytes (2^{10} Gigabytes).

Existen muchas aplicaciones que utilizan la compresión con distintas expectativas: compresión de datos, vídeo y voz, compresión en tiempo real, etc. A su vez, cada uno de estos usos requiere unas características de velocidad, reversibilidad (que el algoritmo pueda ser aplicado de forma reversible para obtener los datos originales), pérdida mínima de información (en el caso de los algoritmos con pérdida de información, etc.) (Salomon, 2002).

Métodos de Compresión.

Para la compresión de datos se ha elegido una clasificación muy general tomando como característica de división la reversibilidad del algoritmo y puede dividirse en dos tipos principales: compresión con pérdida (*lossy*) y compresión sin pérdida (*lossless*).

- Compresores ***lossless*** o sin pérdidas, en el sentido de que guarda absolutamente toda la información original (es reversible). Se utilizan para la compresión de datos, en los que no se puede dar pérdida de información.
- Compresores ***lossy*** o con pérdidas. La compresión hace que se pierda información de la fuente original. Sin embargo, esta pérdida es insignificante en comparación con la ganancia en compresión. Se utiliza sobre todo en imágenes y sonido, donde se puede "engañar" a los sentidos y una pérdida de calidad apenas es percibida (pero ocasiona un radio de compresión mucho mayor).

1.4.1 Compresión Lossless.

Los compresores ***lossless*** se caracterizan porque la tasa de compresión que proporcionan está limitada por la entropía (redundancia de datos) de la señal original. Los métodos de codificación sin pérdidas se utilizan sobre todo en aplicaciones de codificación de datos binarios de aplicaciones informáticas en los que es absolutamente necesario recuperar la información original. Los formatos más populares son el **zip** y el **arj**. En tratamiento de imágenes los métodos de compresión sin pérdidas encuentran su aplicación en la codificación de imágenes médicas o científicas en las que puede resultar crítica la pérdida de parte de la información. Entre estas técnicas destacan las que emplean métodos estadísticos, basados en la teoría de Shannon, que permite la compresión sin pérdida. Por ejemplo: codificación de Huffman y codificación aritmética. Son métodos idóneos para la compresión dura de archivos (Fowler, 1994).

Hay básicamente dos tipos de algoritmos de compresión *lossless*:

- Compresores estadísticos.
- Compresores basados en diccionario ó sustitucionales.

Métodos Estadísticos: Los métodos de compresión estadísticos usan las propiedades estadísticas de los datos que van a comprimirse para asignar códigos de longitud variable a los símbolos individuales en los datos. Existen varios métodos estadísticos propuestos, la principal diferencia entre ellos es la forma en la que cada uno obtiene las probabilidades de los símbolos. Estos emplean un modelo para obtener las probabilidades de los símbolos, la calidad de la compresión que se logra depende de que tan bueno sea ese modelo. Para obtener buena compresión, la estimación de la probabilidad se basa en el contexto en el que ocurren los símbolos. Dadas las probabilidades de los símbolos, la codificación se encarga de convertir dichas probabilidades en una cadena de bits para obtener los datos en forma comprimida. Los métodos estadísticos más comúnmente usados son la Codificación aritmética (Witten, 1987), la Codificación Predictiva (Cleary, 1984) y la Codificación Huffman (Huffman, 1952).

Codificación Aritmética: asigna un único código a cada posible conjunto de datos. Se basa en las probabilidades de ocurrencia de los mensajes emitidos por la fuente de información y en la representación de un valor del intervalo [0,1] con más decimales (más precisión) cuanto más información contengan los datos a comprimir (Witten. I.H., 1999).

Codificación Predictiva: Procuran *predecir* el siguiente mensaje de la entrada tomando como base de conocimiento la entrada procesada hasta ese momento (en el fondo, también probabilidades). Si el mensaje que se encuentra en la entrada coincide con el predicho, su codificación se podrá hacer con menos bits. Si no, su codificación se hará con más bits, que permitirán entonces sincronizar al descompresor para que mantenga sus tablas internas idénticas a las del compresor sin pasárselas explícitamente(Raita, 1987).

Codificación Huffman: La idea es asignar códigos binarios lo más cortos posibles a aquellos símbolos que ocurren con mayor frecuencia en los datos. Los símbolos con poca frecuencia tendrán asignado códigos binarios de longitud más grande. El óptimo desempeño del algoritmo se consigue cuando el número de bits asignado a cada carácter es proporcional al logaritmo de la probabilidad de mismo. Inicialmente este algoritmo fue

construido para ficheros textos y hoy en día es muy usado para imágenes y transmisión de mensajes. A partir de los caracteres ASCII se hizo un estudio de los más usados y teniendo en cuenta la probabilidad de ocurrencia de éstos se fue asignando representaciones binarias a ellos. Los códigos más cortos fueron asignados a los valores más usados y los más largos a los menos comunes. Tratando de ser general a la hora de hablar del código asignado a un carácter o a un mensaje, o a un pedazo de código se le denominará, en lo sucesivo, elemento. En el caso más simple, el alfabeto de salida en el que se realiza la codificación es binario. Esto quiere decir que de cada nodo partirán dos ramas, una para el 0 y otra para el 1. El código para cada elemento se construye siguiendo el camino desde el nodo raíz hasta la hoja que representa el elemento.

Métodos basados en diccionario: Usan el principio de remplazar cadenas de datos con *codewords* que identifican a esa cadena dentro de un diccionario. El diccionario contiene una lista de subcadenas y *codewords* asociados a cada una de ellas. El diccionario que contiene las cadenas de símbolos puede ser estático o adaptable (dinámico). Estos métodos a diferencia de los estadísticos, usan *codewords* de longitud fija y no requieren de las estadísticas de los símbolos para realizar la compresión. Teniendo en cuenta que las cadenas pueden ser arbitrariamente largas, la producción del *codeword* en lugar de la cadena representa un ahorro de información y por lo tanto compresión. Prácticamente, todos los métodos de compresión sustitucionales están basados en los métodos de compresión desarrollados por Jacob Ziv y Abraham Lempel en los 70s, los métodos LZ77 (Ziv, 1977) y LZ78 (Ziv, 1978). Incluidos dentro de este grupo consideraremos a los algoritmos RLE, que posee un diccionario de longitud 1 byte, y el LZW (Welch, 1984) entre otros.

1.4.2 Compresión Lossy.

Los métodos de compresión con pérdida de información logran alcanzar unas tasas de compresión más elevadas a costa de sufrir una pérdida de información sobre la imagen original. Por ejemplo: JPEG, compresión fractal, EZW, SPIHT, etc. El proceso de

codificación para este tipo de compresión no necesita ser reversible. Cuando se comprime un fichero, se transmite, y cuando de descomprime, el usuario espera conseguir el original, igual hasta el último bit. En la transmisión de información, ya sea audio, imagen o sonido; este requerimiento no existe. Para la compresión de imágenes se emplean métodos *lossy*, ya que se busca alcanzar una tasa de compresión considerable, pero que se adapte a la calidad deseada que la aplicación exige sobre la imagen objeto de compresión.

El principio general sobre el que se sustenta la codificación con pérdidas es que no resulta necesario codificar aquellas componentes de la información que no son observables por los sistemas de percepción humana. Por lo tanto, estos métodos se fundamentan en las características psicofisilógicas de los sistemas auditivo y visual, que son, en última instancia, los que deben evaluar la calidad del algoritmo de compresión. Por ello, es fundamental comprender las limitaciones y características de estos sistemas de percepción para diseñar codificadores en los que las pérdidas de información resulten poco evidentes o incluso inapreciables. La principal ventaja de estas estrategias de codificación es que consiguen unos factores de compresión muy superiores a los que se obtienen con los métodos sin pérdidas. Además, suelen ser métodos escalables con la aplicación, es decir, el grado de pérdida de calidad que se tolera depende del ámbito al que se destine el codificador.

1.5 Técnicas de Clasificación.

Los problemas de clasificación aparecen en prácticamente todos los dominios de aplicación, por eso la solución a esta clase de problemas se ha enfrentado desde diferentes disciplinas. Un factor importante para decidir desde que perspectiva abordarlo es el tipo de conocimiento necesario para realizar la clasificación, el cual puede ser estructurado y no estructurado; en el primer caso contamos con un algoritmo que nos permite realizar la clasificación del evento u objeto a partir de su descripción, en el segundo caso no contamos con esa vía de clasificación. En esta última alternativa es

donde las técnicas de IA son aplicables y necesarias para poder construir un procedimiento o modelo de clasificación.

Clasificar involucra necesariamente el concepto de conjunto; de manera general el proceso de clasificación consiste en realizar una separación de los elementos de un conjunto *C* en diferentes subconjuntos C_i, $i = 1, \ldots, P$, denominados clases, con base en la medición de las características que poseen los elementos de *C*. Una vez que se determinan las propiedades de los subconjuntos en los que se clasificará al conjunto original (modelos), los elementos de éste son comparados con cada uno de los modelos, para establecer a cual de ellos pertenecen.

Diferentes técnicas de la IA se han considerado en esta problemática, entre ellas están: las máquinas de soporte vectorial (Burges, 1998), los Árboles de Decisión (Crossland, 1995), los Sistemas Basados en Casos (Plaza, 1994), y las Redes Neuronales Artificiales (Hilera González, 1995) son ejemplos de estas técnicas.

Máquina de soporte vectorial (*Support Vector Machine*, SVM): es otra técnica de aprendizaje supervisado, la cual parte de la teoría de aprendizaje estadístico y se basa en el principio de minimización de riesgo estructural. Concretamente, fundamenta las decisiones de clasificación, no basadas en todo el conjunto de datos, sino en un número finito y reducido de casos, que constituyen los "vectores soporte". Puede dividirse en SVM lineal y no lineal (basado este último en funciones núcleo) (Vapnik, 1995). Actualmente hay muchas aplicaciones que utilizan las técnicas de las SVM como por ejemplo las de OCR por la facilidad de las SVMs de trabajar con imágenes como datos de entrada (Betancourt, 2005).

Árbol de Decisión: Son herramientas excelentes para ayudar a realizar elecciones adecuadas entre muchas posibilidades. Su estructura permite seleccionar una y otra vez diferentes opciones para explorar las diferentes alternativas posibles de decisión. Los árboles de decisión son guías jerárquicas multi-vía donde la jerarquía se refiere a que la

toma de una decisión o camino lleva a otra, hasta que todos los factores o características involucradas se hayan tomado en cuenta. Un árbol de decisión lleva a cabo un test a medida que este se recorre hacia las hojas para alcanzar así una decisión. Un nodo hoja representa el valor que devolverá el árbol de decisión. y finalmente la ramas brindan los posibles caminos que se tienen de acuerdo a la decisión tomada (Quintero Marcela, 2006).

Sistemas Basados en Casos: son sistemas que implementan como método de solución de problemas el Razonamiento Basado en casos, consiste en resolver problemas recordando situaciones previas similares y reutilizando la información y el conocimiento sobre esa situación. Para poder llevar a cabo la tarea de recuperación es necesario tener un algoritmo de recuperación y una medida de similitud que se usarán para obtener un conjunto de casos similares. La tarea de reutilización esta centrada en dos aspectos: (a) las diferencias entre el caso pasado y el actual, y (b) qué parte o partes del caso recuperado pueden ser transferidas al nuevo caso. En algunos casos la tarea de reutilización se reduce a copiar la solución pasada al nuevo caso, pero en otros casos esta solución no puede ser aplicada directamente y tiene que ser adaptada (García, 1996).

Redes Neuronales Artificiales (RNA): son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un conjunto de elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable llamado peso. La actividad que una unidad de procesamiento o neurona artificial realiza en un sistema de este tipo es simple. Normalmente, consiste en procesar una suma pesada de los valores de las entradas (*inputs*) que recibe de otras unidades conectadas a ella, comparar la evaluación de esta cantidad en una función de activación con su valor umbral respectivo y, si lo iguala o supera, enviar activación o salida (output) a las unidades a las que esté conectada. Tanto las entradas que la unidad recibe como las salidas que envía dependen a su vez del peso o fuerza de las conexiones por las cuales se realizan dichas operaciones (Moreno, 2002).

Como podemos apreciar todas estas tecnologías de la IA son igualmente aceptables en su aplicación a un problema de reconocimiento de caracteres con las peculiares adaptaciones del problema a estas pero actualmente las redes neuronales son las que más destacan entre ellas por su habilidad para tolerar el ruido en la información de entrada, para autoorganizarse y para aprender. Es usada en problemas como el reconocimiento de dígitos escritos a mano, el reconocimiento de voz, el reconocimiento de rostros y el procesamiento de imágenes; ejemplos de estas aplicaciones son: reconocimientos de matrículas de automóviles, reconocimiento de la escritura braille, reconocimiento extraído de planos, neurocomputadores multi-cpu para el reconocimiento de caracteres manuscritos, etc.

1.5.1 Aspectos teóricos sobre Redes Neuronales.

La definición de una red neuronal se puede resumir como una red celular física que es capaz de adquirir, almacenar y utilizar conocimiento experimental, en relación con la eficiencia y capacidad de la red. Se puede formalizar esta definición agregando que una red neuronal puede ser definida como una interconexión de neuronas, tales neuronas poseen un conjunto de entradas y una salida, y un mecanismo propio para responder a sus entradas y producir una respuesta a través de su salida, que a su vez puede estar conectada a la entrada de otras neuronas, incluida ella misma, mediante pesos (Figura 1.3).

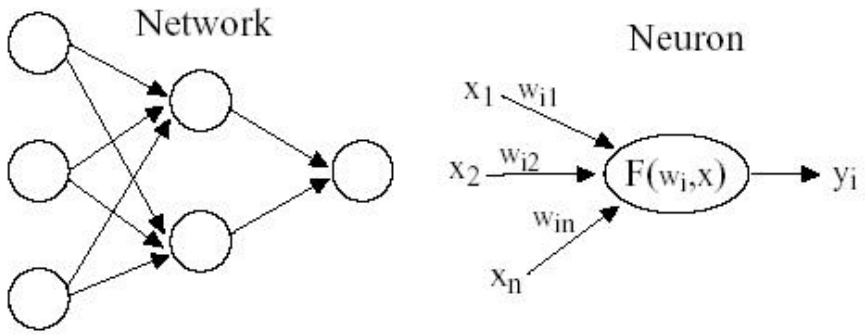


Figura 1.3 Ejemplo de red artificial y neurona artificial.

Toda RNA se caracteriza por tres elementos esenciales: la topología, el aprendizaje y la función de activación.

La topología o arquitectura de una red neuronal consiste en la organización y disposición de las neuronas en la misma, formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de dicha red. En este sentido, los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

Cuando todas las neuronas de la red poseen conexiones sólo con unidades de capas delanteras, se dice que este tipo de red es "hacia delante" o *feedforward*. Sin embargo, en un gran número de estas redes también existe la posibilidad de conectar la salida de las neuronas de capas posteriores a la entrada de capas anteriores; a estas conexiones se las denomina *conexiones hacia atrás o feedforward / feedback.*

Según (Hilera González, 1995), el 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 otras palabras, el aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el mismo se reducen a la destrucción, modificación y creación de conexiones entre las neuronas (Matich, 2001).

En los modelos de RNA, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero. De la misma manera, una conexión se destruye cuando su peso pasa a ser cero. Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por lo tanto, se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen estables ($\frac{dw_{ij}}{dt} = 0$).

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir, que tiene un *"estado de activación*". Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas solamente dos, al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado.

La *función de activación* calcula el estado de actividad de una neurona; transformando la entrada global (menos el umbral, θ_i) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (–1 a 1). Esto es así, porque una neurona puede estar totalmente inactiva (0 o –1) o activa (1).

Dentro de los modelos de redes neuronales más referenciados en la literatura para reconocer patrones se destacan las red neuronal de Hopfield (Hopfield, 1982) y el Multilayer Perceptron (Rumelhart et al., 1986) (Diaz Sardiñas, 2000).

El *Multilayer Perceptron* (Perceptron Multicapa), como su nombre lo indica está conformado por varias capas, en cada una de las cuales, posee uno o más Perceptron. Actualmente se aplican a diferentes clases de problemas como:

- Reconocimiento del habla.
- Reconocimiento óptico de caracteres.
- Reconocimiento de patrones visuales.
- Control de procesos.
- Predicción de series temporales.

Actualmente este modelo de red esta muy difundido en las distintas ramas del conocimiento lo cual podemos comprobar mediante la diversidad de problemas que requieren su uso de acuerdo a sus peculiaridades:

1. Sistema de generación automática de redes neuronales digitales para FPGA que predice variables climáticas (temperatura, humedad del suelo, ventilación,…) en un medioambiente acotado con microclima propio.

2. Diseño de algoritmos que calculen el precio de una opción de compra de compañías telefónicas utilizando datos históricos del mercado, a partir de variables de corte financiero (precios, volúmenes, intereses, tiempos de vencimiento,….).
3. Construcción de herramientas estadísticas no paramétricas para la clasificación donde se obtienen mejores resultados frente a modelos de regresión logística y análisis discriminante.
4. Detección de arritmias ventriculares, filtrado adaptivo de bioseñales, clasificación de patrones del ritmo cardiaco e identificación de contracciones ventriculares anormales mediante datos electrocardiográficos de una base de datos de arritmias cardíacas pretendiéndose clasificar los latidos ventriculares en normales y anormales.
5. Interfaces de usuario para el reconocimiento eficiente de caracteres alfanuméricos donde los caracteres pueden presentarse en una gran variedad de tamaños, tipos, inclinaciones, o incluso en cursiva o subrayados.

La RNA de Hopfield es una red monocapa, puede trabajar con varias neuronas y permite conexiones recurrentes pero siempre dentro de la misma capa. Es un modelo de red con el número suficiente de simplificaciones como para poder extraer analíticamente información sobre las características relevantes del sistema. Presenta un tipo de aprendizaje no supervisado *off-line* de tipo Hebbiano, entonces existen dos etapas: la etapa de aprendizaje y la de funcionamiento de la red. Este tipo de aprendizaje asegura la estabilidad de la red, es decir la convergencia hacia una respuesta estable cuando se presenta una información de entrada (Freeman, 1993). Entre las aplicaciones fundamentales aparecen:

1. **Reconocimiento de imágenes y voz:** Puede reconstruir versiones distorsionadas o parciales de imágenes almacenadas en la fase de aprendizaje. La limitación de la red es la imposibilidad de recuperar imágenes rotadas o trasladadas.

2. **Resolución de problemas de optimización complejos:** resuelve estos problemas que con algoritmos clásicos el tiempo de resolución es de crecimiento ilimitado. Por ejemplo, colocar *N* reinas en un tablero de *N*x*N*.
3. **Desarrollo de memorias direccionadas por contenido**: Los elementos de la memoria no estarían ordenados según índices numéricos, sino según parte de su contenido.
4. **Brindar soporte a la toma de decisiones:** En el ámbito del otorgamiento de créditos bancarios se estudia la información disponible sobre ciertas características de un cliente para determinar si se le otorga o no el crédito.
5. **Reconocimiento de patrones:** Se han desarrollado *frameworks* para la clasificación de patrones surgido en el contexto académico el cual realiza estudios de los elementos comunes y las variabilidades existentes en sistemas análogos.

Entre otros de sus usos debido a sus capacidades para "recordar" podemos mencionar:

- Modelación neurobiológica.
- Tareas lingüísticas.
- Control de procesos dinámicos.

En nuestros días se experimenta en el desarrollo de OCRs que utilicen las ventajas que brindan los diversos modelos de redes neuronales. La justificación de tal afirmación está dada por la existencia de un reconocedor de caracteres deformados por impresiones, que combina tanto las redes neuronales de Hopfield como la MLP, desarrollado en universidades de Francia y Argelia por un grupo de docentes. La idea es hacer un procesamiento inicial con Hopfield para hacer una restauración del carácter y luego dicha salida sería la entrada a la red MLP que decidiría la clasificación final. Los resultados alcanzados en la investigación superaron el 99% de la clasificación correcta para diferentes niveles de ruidos presentados, los cuales superaron a los alcanzados por tales modelos de manera independiente (Namane, 2005).

Conclusiones parciales.

En los sistemas de reconocimiento de caracteres existen tres procesos esenciales: Representación, Generación de una Base de Conocimientos e Identificación– Clasificación. De la calidad de cada uno de ellos dependerá la calidad del reconocedor que se alcance.

De los métodos de segmentación analizados resulta el Método Basado en Píxeles el más conveniente dado porque sólo se necesita el valor de gris de un píxel y fijar umbrales para estos para así identificar las objetos de interés del resto pasando por alto el evitar cambiar el tamaño del objeto puesto que esta fijado por un numero de píxeles así como los aislamientos que puedan existir de regiones no conectadas a las de interés.

Entre los compresores abordados hacemos mayor énfasis en la codificación Huffman y RLE, ya que de acuerdo a sus características en la reducción de la información así como su inclusión en la compresión estándar JPEG se hace necesario su uso para la descompresión de la información que acarrean los archivos imágenes de tal formato.

Varias técnicas de IA pudieran ser utilizadas como clasificadores o reconocedores de caracteres, se recomienda el modelo de Hopfield dado por su capacidad de reconstruir versiones distorsionadas o parciales de imágenes almacenadas en la fase de aprendizaje.

Capítulo II: "Diseño e Implementación de un OCR numérico para ETECSA."

En este capítulo se diseña e implementa un reconocedor utilizando un modelo de red neuronal Hopfield discreto a partir de la foto tomada de los metro-contadores y transformando el formato JPG a una imagen binaria.

2.1 Planteamiento del problema.

En la Gerencia de Holguín hasta el año 2003 las fotos se tiraban con cámara que utilizaban rollos fotográficos, los cuales había que revelar y luego imprimir cada una en papel fotográfico. Este servicio de revelado e impresión se realizaba con un tercero, al cual se le entregaba los rollos y el papel y estos cobraban 4.00 en MN por cada foto.

Por otra parte la calidad de la cámara, la tirada, así como el relevado no era la mejor lo que atentaba cuantitativamente en errores producidos en la captación, así como lecturas ilegibles que se debían igualar implicando que en el mes no se facturara lo que realmente se había consumido por el cliente.

En el propio año 2003 se comenzaron a distribuir cámaras digitales para que los Centros de Facturación realizaran pruebas, en la Filial TISW se realizaron las mismas, se le hicieron adaptaciones a la cámara y al embudo y se logró que las fotos tiradas tuviesen la calidad requerida.

En vista a lo anterior y según la actualización del Banco de Problemas de la Filial se consideró la idea de hacer un software que permitiera la captación por medio de visualización directa en la pantalla y que permitiera ampliar las mismas y focalizar a la vez la lectura de un metro en específico. Este software debía ser flexible en cuanto a la configuración y parametrización para su utilización en todas las Gerencias Territoriales del país.

ETECSA tiene varias centrales analógicas, y en cada una de ellas varios metros contadores que deben ser leídos para realizar el proceso de facturación de cada cliente. Como promedio cada pizarra tiene 100 contadores y existen 4 pizarras en un municipio y 60 en una provincia. Existe un operario que registra los números y para ello dedica varias horas de trabajo, con el consiguiente cansancio de sus ojos. Además no se puede garantizar una adecuada seguridad en los servicios, evitando cualquier uso indebido de la información. Estos elementos provocan la necesidad la automatización del proceso y por consiguiente ayudarán a evitar errores en la trascripción de los números, agilizándose este proceso de lectura.

Es por ello la necesidad de aumentar la funcionalidad del software ya construido, agregándole un módulo que permita reconocer el valor captado de un metro contador una vez que se haya visualizado en la pantalla y cuya solución reside en la implementación de un OCR.

2.2 El Estándar JPEG.

El primer elemento imprescindible de este proyecto lo constituye la imagen obtenida por la cámara digital que utilizan los especialistas que se desempeñan en tal tarea y a partir de ello su formato resulta de vital interés para el procesamiento que se llevará a cabo más tarde. Entre unos de los formatos más habituales se encuentra el JPEG (Joint Photographic Expert Group). Este formato se caracteriza principalmente por ser abierto, los derechos de autor son libres y puede ser usado o implementado en un programa, sin necesidad de pagar por derechos de autor.

JPEG o JPG nace como una respuesta a las limitaciones de otros formatos, se utiliza como extensión predeterminada por las cámaras digitales debido a que permite comprimirlas sin necesidad de bajar su calidad en la resolución y nos libera el espacio.

Cabe mencionar, que JPEG es un formato de compresión con pérdida de información, es decir que cuando guardamos una fotografía con esta extensión, la información que

contiene la imagen reduce, pero este detalle no es susceptible al ojo humano, porque la calidad de las imágenes sigue siendo alta (Henst, 1997).

Lo que hace a JPEG completamente diferente de otros formatos (PNG, GIF, TIFF,...) es el hecho de aceptar la pérdida de información a la hora de comprimir. Este tipo de compresión consta de 3 pasos.

- Primero se pasa la imagen del formato RGB al formato YIQ. El formato de color YIQ representa una división entre la luminosidad (cantidad de luz percibida) y la información sobre el color. El ojo humano es mucho más sensible a la luminosidad que al color, cosa que se aprovecha para la compresión.
- Después, se realiza una transformación en la imagen mediante la transformada discreta del coseno (TDC), que es una herramienta matemática que permite comprimir la imagen según determinados patrones de frecuencia. Se trata de un método de compresión con pérdida de datos.
- Por último, se codifica el conjunto de datos obtenidos al aplicar la TDC, usando un método que no producen pérdida (código de Huffman).

Para recuperar la imagen se usa el proceso inverso de transformación de imágenes.

El primer paso de la codificación de una imagen con JPEG es la **preparación de bloques** de 8×8.

El segundo paso de JPEG es aplicar una **Transformación Discreta de Cosenos** (Wallace, 1991)**.**

Una vez completado el DCT, llega el tercer paso del JPEG, llamado **cuantización**, en el cual se eliminan los coeficientes DCT menos importantes.

El cuarto paso reduce el valor (0,0) de cada bloque (el de la esquina superior izquierda) reemplazándolo por la cantidad en que difiere del elemento correspondiente en el bloque previo (Figura 2.1). Como estos elementos son la media de sus respectivos bloques, deberían cambiar lentamente, así que tomar los valores diferenciales debería reducir la

mayoría de ellos a valores pequeños. No se computan diferenciales de los otros valores. Los valores (0,0) son referidos como las componentes DC; los otros valores son los componentes AC.

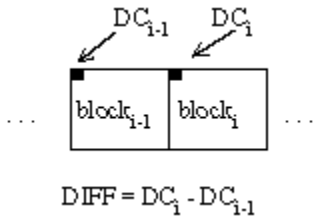


Figura 2.1 Diferencia de los bloques DC obtenidos después de aplicar la DCT.

El quinto paso ubica los 64 elementos de manera lineal y aplica **RLE** a la lista. Recorrer el bloque de izquierda a derecha y de arriba abajo no concentrará los ceros juntos, así que se usa un patrón de recorrido en zig-zag (Figura 2.2).

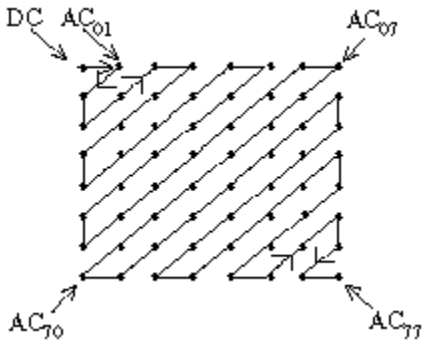


Figura 2.2 Recorrido zig-zag.

Ahora se tiene una lista de números que representan la imagen (en el espacio transformado). El sexto paso codifica, utilizando Huffman, los números de las matrices.

Decodificar una imagen JPEG requiere ejecutar el algoritmo al revés. A diferencia de otros algoritmos vistos, JPEG es muy simétrico: decodificar lleva tanto como codificar.

Es bastante interesante que debido a las propiedades matemáticas de la DCT, es posible llevar a cabo ciertas transformaciones geométricas (p.e. rotación de imagen) directamente en la matriz transformada, sin regenerar la imagen original (Wallace, 1991).

2.2.1 Obtención de la imagen binaria a partir de un fichero JPG.

Este formato de imagen tiene en su encabezamiento la información necesaria para su lectura, pero a diferencia de éste, está dividido en markers y el comienzo de cada uno es designado por el byte FF (en hexadecimal) y según el tipo de marker será el byte siguiente.

La descompresión del formato JPG (o JPEG) no es más que aplicar el método JPEG en sentido contrario, por lo que constaría de los siguientes pasos:

- Según el código Huffman descomprimir los datos, obteniendo dos conjuntos de componentes (1 DC por cada 63 AC).
- Aplicar el método Run-Length a las componentes AC.
- Analizar las componentes DC y AC por separado aplicándole la Inversa de la Trasformada de Cosenos Discreta (IDCT).
- Proceso de Cuantización.
- Transformación del espacio de color.

Este último paso se sustituyó por el de la construcción de la matriz binaria, pues con la componente **Y** bastó para obtener la información necesaria sobre el fondo y los objetos.

El formato JPG, a partir de imágenes RGB, transforma los datos hacia un espacio de color luminancia/crominancia como YCbCr, YIQ o YUV, que tienen una componente de luminancia y dos de crominancia. Teniendo en cuenta que la componente de crominancia en estos ficheros es una versión en blanco y negro de la imagen, podemos descompactar sólo esta componente, pues en este caso no se necesitan los colores. Se van descompactando las componentes de luminancia con las tablas de Huffman, obteniéndose

los coeficientes DC y AC de cada bloque. Como fue mencionado antes, este formato consta de un conjunto de markers, entre ellos uno que define las tablas de Huffman con las que fue codificado. A partir de estas tablas se puede almacenar la información en árboles para facilitar la búsqueda del código asignado a cada símbolo, y nos permita, por el análisis bit a bit, ir decodificando los datos. Existen dos tipos de tablas Huffman, una para las componentes AC y otras para las componentes DC. Seguidamente se decuantifica la matriz poniéndola en posición normal (recordar que estaba almacenado en zig-zag), que no es más que la división por los elementos de la matriz que debe aparecer en el encabezamiento de la imagen. Se le aplica el inverso de CDT (ICDT) y se le suma 128 a cada elemento de la matriz. Se debe tener en cuenta la forma en que fueron almacenadas según las Unidades de código Mínimo (MCU). Cada MCU debe tener 4 bloques de 8×8, de forma que se almacenen 4 bloques de luminancia y seguidamente uno de cada tipo de crominancia. En este caso como analizamos sólo la componente del brillo, el MCU tendrá 4 bloques de estas componentes, como se muestra en la figura 2.3. Así se obtiene la imagen en blanco y negro, donde cada píxel tiene como valor su nivel de gris correspondiente y se pasa a establecer un umbral para asignar los valores de la representación binaria.

Figura 2.3 MCU.

Después de un exhaustivo análisis se llegó a la conclusión de que el umbral debía ser establecido dentro de cada bloque en particular, pues en una imagen en blanco y negro las tonalidades de gris dependen también de los píxeles vecinos. Buscando el mayor valor dentro de cada bloque y restándole 100 se obtiene una media entre el fondo y los objetos representados en esa región.

2.3 Identificación de las partes a reconocer.

Después de tener la representación de la imagen se pasa al procesamiento, para la búsqueda y tratamiento de componentes hasta llegar a la estructura mínima que sería lo que representa a un carácter.

En la escritura común (y se ve representado en los formatos textos digitales) existen elementos que organizan y ubican los caracteres, como los renglones y palabras. En nuestro caso disponemos de 2 renglones a lo sumo y una única palabra que constituye el número a reconocer. Para representar cada uno de ellos se definieron clases que almacenan su posición, dimensiones y la información necesaria en cada caso para continuar el procesamiento, y, una vez clasificados, generar un fichero texto que almacene tales resultados.

Para poder leer los ficheros imágenes y realizar la conversión a matriz binaria se construyó una clase denominada ***CImgToMatrix***, que contiene el método necesario para esto (*ReadFileToMatrix*). De acuerdo a las particularidades del formato tratado se creó una clase que pudiera manipularlo y que implementara el método de la clase mencionada anteriormente ***CJpgToMatrix***, Además esta posee las funcionalidades necesarias para poder descomprimir.los ficheros. Estas clases tienen como propiedad la matriz binaria que resulta de la lectura del fichero (Ver Anexo 3).

Se construyó una clase que encerrará todas las funciones necesarias para la manipulación de una matriz (***CMatrix***). Esta clase tiene atributos definidos como el ancho y el alto de la matriz, así como el arreglo de sus elementos (*píxeles*). En esta matriz las coordenadas *x* representan las filas y las *y* las columnas.

Esta clase encierra estos y otros métodos típicos para manipular una matriz cualquiera, pero para facilitar el tratamiento de los datos de la imagen y su conversión a matriz

binaria se creó otra clase que herede de ésta anterior (***CImgMatrix***). La declaración de esta clase es la siguiente:

```
class CImgMatrix: public CMatrix
{
private:
      double m_dRAltoPromedio;
      CObArray m_oaRenglones;
public:
      double GetMatrixCharAnchoPromedio();
      BOOL WriteToText(CString FNameOpened);
      double GetRAltoPromedio();
      double CalculateRAltoPromedio();
      BOOL MatrixToText(CString FNameOpened);
      CImgMatrix(CMatrix & matrix);
      CObArray& GetRenglonesList();
      CImgMatrix();
      CImgMatrix(BYTE* newMatrix, int newHeight, int newWidth);
      virtual ~CImgMatrix();
      int FindRenglones();
      CImgMatrix& operator=(CImgMatrix *otherJpgMatrix);
};
```

Para representar cada una de las componentes se definieron tres clases que almacenan su posición, dimensiones y la información necesaria en cada caso para continuar el procesamiento. Estas clases se denominaron: ***CPosition***, ***CDimention*** y ***CComponent***.

CPosition tiene sólo dos propiedades, la columna y la fila, para la ubicación de éstas componentes en la matriz. *CDimention* se encarga de las dimensiones, por lo que tiene como propiedades un ancho y un largo. *CComponent*, por su parte, consta de una posición y una dimensión. La declaración de estos atributos es:

```
CPosition* m_pPosition; // propiedad de la posición dentro de la matriz.

CDimention* m_dDimention; // dimensiones de la componente.
```

También tiene varios métodos, entre los principales se encuentra *GetMatrixComponent* que me da la submatriz que representa esas dimensiones y esa posición dentro de la matriz principal. También se tienen *CalcHeight* y *CalcWidth* que calculan el ancho y el largo respectivamente dadas las posiciones finales e iniciales de la componente. La declaración de estos métodos es la que sigue:

```
int CalcHeight (CPosition* initPos, CPosition* endPos);

int CalcWidth (CPosition* initPos, CPosition* endPos);

CMatrix GetMatrixComponent (CMatrix* AllMatrix);
```

En nuestro problema como mencionábamos anteriormente tendremos como máximo 2 renglones, considerando el hecho de que alguno de los dígitos de la numeración del metrocontador aparezca incompleto y en tal submatriz queden las partes inferior y superior de 2 dígitos consecutivos. Los CharBox a analizar en la palabra que representa el valor serán tomados con una dimensión fija basada en la medida de los píxeles y cuyo contenido será un digito de los 6 posibles que tiene la palabra. Estas dimensiones forman parte entonces del conjunto de parámetros descriptores que utilizaría la herramienta de clasificación a implementar como entrada. En el próximo capítulo haremos una valoración sobre posibles variantes de clasificadores para nuestro problema y seleccionaremos aquella con mejor desempeño en tal tarea de reconocimiento.

2.4 Clasificación de los caracteres.

Como ya es conocido los caracteres a reconocer serán los números del 0 al 9, los cuales deberán estar definidos por mapas de píxeles (blanco y negro) de 35 x 17, de manera similar a la siguiente figura:

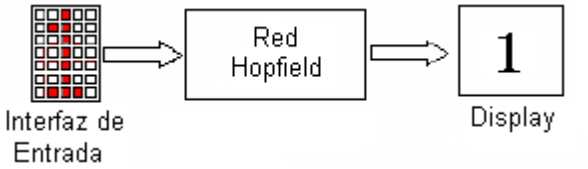


Figura 2.4 Entrada y salida de la red.

Dadas las peculiaridades del sistema que estamos tratando se decidió utilizar como reconocedor el modelo de activación original de Hopfield Discreto para la cual quedaron definidos los siguientes aspectos.

- Los parámetros descriptores serán los valores binarios de los píxeles en la submatriz binaria que corresponde a un caracter (CharBox) los cuales constituyen un vector de igual dimensión que dicha submatriz.
- Para la entrada a la red se considera un vector de 595 componentes que se corresponden con las neuronas de la red, dado que la submatriz que representa el charbox posee dimensiones 35 x 17 píxeles que es la dimensión fija de la región de la numeración del metrocontador anteriormente.
- Los valores de las componentes de dicho vector son 1 y -1 como corresponden con las entradas y salidas del modelo los cuales representan los valores 1 y 0 respectivamente en la submatriz del CharBox.
- El conjunto muestral de patrones a aprender consta de 80 ejemplares distribuidos entre los 10 dígitos de acuerdo a su nivel de dificultad.

2.4.1 Aprendizaje.

Se implementarán para el cálculo de los pesos las siguientes variantes:

- Frecuencia Absoluta:

$$w_{pq}^{1} = \frac{\sum_{i=1}^{N} x_{ip} * x_{iq}}{N} \qquad (2.1)$$

- Frecuencia Relativa:

$$w_{pq}^{2} = \frac{\sum_{i=1}^{N} x_{ip} * x_{iq}}{\sum_{i=1}^{N} x_{ip}} \quad (2.2)$$

- Coeficiente de Correlación de Pearson:

$$w_{pq}^{3} = \frac{N\sum_{i=1}^{N} x_{ip} * x_{iq} - \sum_{i=1}^{N} x_{ip} * \sum_{i=1}^{N} x_{iq}}{\sqrt{\left[N\sum_{i=1}^{N} x_{ip}{}^{2} - \left(\sum_{i=1}^{N} x_{ip}\right)^{2}\right]\left[N\sum_{i=1}^{N} x_{iq}{}^{2} - \left(\sum_{i=1}^{N} x_{iq}\right)^{2}\right]}} \quad (2.3)$$

Al aplicar las dos primeras variantes de cálculo, el valor del peso oscila entre 0 y 1, mientras que el coeficiente de correlación de Pearson permite que este valor fluctúe entre –1 y 1. Estadísticamente se considera como una métrica mucho más acertada que las dos primeras, aunque pudiera ser conveniente el uso de las dos variantes de frecuencia para ciertos casos específicos.

2.4.2 Arquitectura.

Red monocapa con N neuronas cuyos valores de salida son binarios: -1/+1(Figura 2.4).

Cada neurona de la red se encuentra conectada a todas las demás (conexiones laterales), pero no consigo misma (no existen conexiones autorecurrentes). Los pesos asociados a las conexiones entre pares de neuronas son simétricos, lo que significa que el peso de la conexión de una neurona *i* con otra neurona *j* es de igual valor que el de la conexión de la neurona *j* con la neurona *i* (wij = wji).

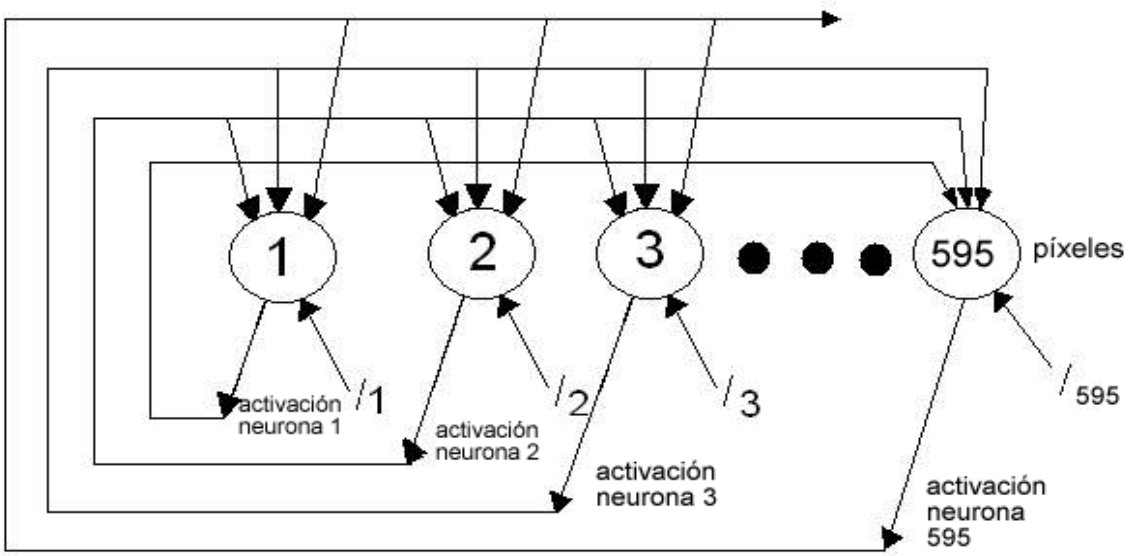


Figura 2.5 Topología de la Red de Hopfield.

La versión discreta de la red fue ideada para trabajar con valores binarios -1/+1, pero mediante un ajuste de los pesos puede utilizarse en su lugar los valores 1/0. Por tanto la función de activación f(x) de cada neurona *i* es de tipo escalón (Figura 2.5):

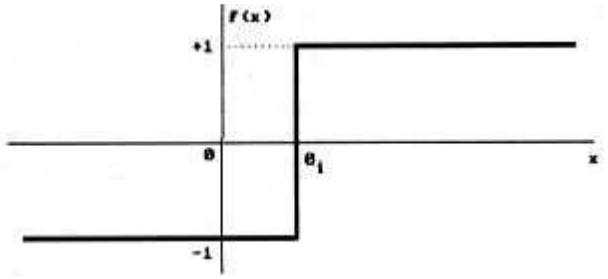


Figura 2.5 Función de activación escalón.

Cuando el valor de x es igual a θi, la salida de la neurona *i* permanece con su valor anterior. θi es el umbral de disparo de la neurona *i*, que representa el desplazamiento de la función de transferencia a lo largo del eje de ordenadas. Suele adoptarse para este modelo un valor proporcional a la suma de los pesos de las conexiones de cada neurona con el resto:

$$f(x) = \begin{cases} +1 & x > \theta_i \\ -1 & x < \theta_i \end{cases} \qquad \theta_i = K \sum_{i=1}^{N} W_{ij} \qquad (2.6)$$

2.4.3 Funcionamiento.

Si consideramos el caso de una neurona concreta de la red, esta neurona recibiría como entrada las salidas de cada una de las otras neuronas, valores que inicialmente coincidirán con los de entrada, multiplicadas por los pesos de las conexiones correspondientes. La suma de todos estos valores constituirá el valor de entrada neta de la neurona, al que le será aplicada la función de transferencia, obteniéndose el valor de salida correspondiente: -1/+1 ó 0/1 si la red es discreta, y un número real en el rango [-1,+1] ó [0,1] si la red es continua.

$$y_k(t+1) = \text{sgn}\left(\sum_{j=1}^{595} w_{jk} * y_j(t)\right) \quad \text{para} \quad k = 1,...,595 \tag{2.7}$$

Donde ***t*** representa la iteración o momento y ***k*** una neurona específica. Dicho proceso concluye cuando las salidas en cada iteración dejen de variar. El modo de operación de la red utilizado en la actualización de las neuronas puede ser tanto el paralelo como el secuencial.

La descripción anterior correspondería a un primer paso en el procesamiento realizado por la red. Este proceso continúa hasta que las salidas de las neuronas se estabilizan, lo cual ocurrirá cuando dejen de cambiar de valor. Entonces, el conjunto de estos *(N)* valores de salida de todas las neuronas constituye la información de salida que ha generado la red, que se corresponderá con alguna de las informaciones que durante la etapa de aprendizaje fueron almacenadas en la misma.

Se implementa funcionamiento paralelo ó síncrono, es decir, que las salidas son generadas por las neuronas al mismo tiempo en cada iteración y con funcionamiento secuencial ó asincrónico, es decir, que la salida a la cual converge la red puede ser diferente en función del orden de la secuencia de activación de las neuronas.

La salida final de la red una vez logrado un estado estable es procesada por funciones de similitud correspondiéndose entonces con uno de los patrones aprendidos para lograr una clasificación. La función de similitud analizada fue la distancia de Manhattan. Existen otras métricas que nos permiten al igual que las anteriores la comparación entre el patrón de salida y el deseado pero en nuestro problema la mayoría de ellas no aportan resultados pues su desempeño es análogo al de aquellas seleccionadas (Anexo 4).

2.5 Implementación de la Red

La funcionalidad de la misma está dada por la clase ***CHopfiledNet*** en los ficheros fuentes Hopfield.cpp y Hopfield.h la cual posee un conjunto de definiciones básicas para su desempeño tales como transformaciones de los valores binarios de los píxeles a los valores binarios que serán la entrada al sistema, el número de filas y columnas de la matriz de neuronas que nos da su número exacto entre otras.

```
#define LO            -1
#define HI            +1
#define BINARY(x)     ((x)  == LO ? FALSE : TRUE)
#define BIPOLAR(x)    ((x) == FALSE ? LO : HI)
#define X             35         /* número de filas */
#define Y             17         /* número de columnas */
#define N             (X * Y)    /* número de neuronas */
```

La declaración de la clase para la red es la siguiente:

```
class CHopfieldNet
{
     DECLARE_DYNCREATE(CHopfieldNet);
// Attributes
private:
    INT    Units;     /* - number of units in this net         */
    INT*   Output;    /* - Output of ith unit                  */
    INT*   Threshold; /* - threshold of ith unit               */
    INT**  Weight;    /* - connection weights to ith unit      */
```

```
    INT   Input [NUM_DATA][N];/*-values of units of ith pattern*/
// Operations
public:
      CHopfieldNet();             /* Constructor */
      virtual ~CHopfieldNet();    /* Destructor */
      void CalculateWeights();    /* Entrenamiento con suma pesada
                                  */
      void CalculateWeightsFrecAbs();  /* Entrenamiento con
                                       frecuencia absoluta  */
      void CalculateWeightsFrecRel();  /* Entrenamiento con
                                       frecuencia relativa */
      void CalculateWeightsCorrPearson();   /*Entrenamiento    con
                                       correlación de pearson*/
      void SetInput(INT* input);  /* Entrada del patron */
      BOOL PropagateUnit(INT i);  /* Activación de la neurona i
                                  modo secuencial*/
      BOOL PropagateUnitParallel(INT i, INT* OldOutput);
                                  /* Activación de la
                                      neurona i modo parelelo */
      void PropagateNet();        /* Explotación de la red modo
                                  secuencial */
      void PropagateNetParallel();     /* Explotación de la red
                                       modo  paralelo */
      void InitializeNetwork();   /* Inicialización de la red */
      INT  Similitude(INT* input);     /* Función de asociación
                                       por semejanza */
      INT  DiscreteEnergy();      /* Función de energía de
                                  Hopfield */
};
```

Como se puede apreciar estos métodos resumen el procesamiento del algoritmo que sigue la red con solo leer sus nombres, ejemplo: inicializar la red, calcular pesos, capturar entrada, propagar unidad que no es más que calcular la activación de una neurona en una iteración, propagar la red tanto secuencial como paralela, que es el cálculo de todas las activaciones de las neuronas hasta llevarlas a un estado estable, y por último, funciones para comparar la información aprendida con la salida final para llegar a una decisión en la clasificación.

Conclusiones parciales

El formato JPG resulta idóneo para la representación de la imagen debido a la magnitud del número de imágenes que se procesan mensualmente en la Gerencia que asciende a más de 60 como promedio. Por esta razón este formato permite almacenar gran número de fotos en un espacio sensiblemente menor que otros formatos de imágenes con la debida compresión de información, para reducir así su tamaño, pero con la consiguiente degradación de la calidad de la imagen; lo cual se valora poco para la tarea en cuestión que se pretende.

Se diseñó e implementó un modelo de Holpfield discreto. La implementación incluye varias formas de calcular los pesos de la RNA y funcionamiento secuencial asíncrono y funcionamiento paralelo síncrono.

La implementación es sencilla y asequible, fácil de modificar de acuerdo a las especificaciones que se requieran en elementos de la red como algoritmos de cálculo de pesos y similitudes.

Capítulo III: "Evaluación y Modo de Uso del MetroSimulador."

Se valida el clasificador implementado y se explican las bondades del software. Finalmente se describe el manual de usuario.

3.1 Evaluación del MetroSimulador

Los problemas de clasificación incluyen como un aspecto muy importantes la forma de evaluación a utilizar para tener una medida de cuan bien funcionan, y a partir de ahí obtener aquellas dificultades que le imposibiliten realizar su tarea correctamente para trabajarlas en función de mejorar su rendimiento.

Sin lugar a dudas, la métrica más conocida y empleada en la obtención del desempeño de una RNA es el por ciento de clasificaciones correctas. Esta métrica es la que utilizan los sistemas conexionistas multicapas como MLP y otros, los cuales han logrado una probada eficiencia en la resolución de problemas de clasificación.

En el campo de la búsqueda asociativa de información, donde hay bases de conocimiento cuyos atributos no han sido particionados a priori en predictores y objetivos, o donde inclusive el rasgo clase puede modelarse como borroso, la métrica antes descrita resulta insuficiente para dar una idea del desempeño global de la RNA asociativa.

Por ello, es preciso buscar una métrica más general que pueda ser utilizada ante cualquier tipo de problema que se nos presente. Básicamente, la idea de la métrica utilizada es calcular la diferencia que existe entre las activaciones de las neuronas correspondientes a los rasgos objetivos de los vectores P y $\overline{P}$, quienes a su vez representan la salida deseada y la obtenida, respectivamente, para cada patrón de la muestra de control.

3.1.1 Aspectos sobre la evaluación en la clasificación.

No existe un modelo de clasificador mejor que otro de manera general; para cada problema nuevo es necesario determinar con cuál se pueden obtener mejores resultados, y es por esto que han surgido varias medidas para evaluar la clasificación y comparar los

modelos empleados para un problema determinado. Las medidas más conocidas para evaluar la clasificación están basadas en la matriz de confusión que se obtiene cuando se prueba el clasificador en un conjunto de datos que no intervienen en el entrenamiento. A continuación se muestra la matriz de confusión de un problema de dos clases, donde C_1 es la clase negativa y C_2 la clase positiva:

	Clase obtenida	
Clase Real	C_1	C_2
C_1	TN	FP
C_2	FN	TP

Figura 3.1 Matriz de confusión.

Donde, TP y TN son los elementos bien clasificados de la clase positiva y negativa respectivamente. FP y FN son los elementos negativos y positivos mal clasificados respectivamente. Basados en estas medidas, se calcula el error, la exactitud (*accuracy*), la razón de TP (*TP rate*) o sensitividad, la razón de FP (*FP rate*), la precisión o especificidad, que se dan por las expresiones siguientes:

$$Error = \frac{FP + FN}{TP + TN + FP + FN} \tag{3.1}$$

$$Exactitud = 1 - Error \tag{3.2}$$

$$Razón\ de\ TP = Sensitividad = \frac{TP}{TP + FN} \tag{3.3}$$

$$Razón\ de\ FP = \frac{FP}{FP + TN} \tag{3.4}$$

$$Precisión = Especificidad = \frac{TP}{TP + FP} \tag{3.5}$$

Otra forma de evaluar el rendimiento de un clasificador es por el análisis de la llamada *Receiver Operator Curve* (ROC) (Provost, 1997). En esta curva es representado el valor de razón de TP vs. la razón de FP, mediante la variación del umbral de decisión. Se denomina umbral de decisión a aquel que decide si una instancia *x*, a partir del vector de salida del clasificador, pertenece o no a cada una de las clases. Usualmente, en el caso de dos clases se toma como umbral por defecto 0.5; pero esto no es siempre lo más conveniente. Se usa el área bajo esta curva, denominada AUC (*Area Under the Curve*) como un indicador de la calidad del clasificador. En tanto dicha área esté más cercana a 1, el comportamiento del clasificador está más cercano al clasificador perfecto (aquel que lograría 100% de TP con un 0% de FP).

Pero estas medidas no son suficientes, usándolas así simplemente, para evaluar un clasificador. La forma de dividir los datos en conjunto de entrenamiento y prueba es también muy importante. Existen varias técnicas para esto, la más vieja es *método R* (*resubstitution*) el cual se basa en entrenar y probar el clasificador con la misma base de datos. Este método puede traer como consecuencia un sobre-aprendizaje del clasificador, o sea que el clasificador más que generalizar el conocimiento de los datos, aprenda esto "de memoria". Otro método usado es el *método H* (*Hold-out*), que divide la base a la mitad, una mitad para entrenamiento y otra para prueba. Una versión de éste es el *data shuffle* que realiza *n* veces el *método H* y promedia los resultados. *Validación cruzada k veces* (*k-fold cross-validation*) es uno de los más usados, este método se basa en dividir la base en *k* pedazos y realizar *k* procesos de entrenamientos y pruebas, donde el proceso *i* se basa en tomar el pedazo *i* para prueba y el resto para entrenamiento. Sea *I* la cantidad de instancias de la base de datos, si *k=I* entonces el método se denomina *leave-one-out*. El método *Boostrap* se basa en la generación de *n* conjuntos de cardinalidad *I* desde el conjunto de datos original, con reemplazo.

Actualmente se usa también en vez de dos conjuntos de datos, tres: uno para entrenamiento, uno para prueba y un tercero para validación. Este último se usa como seudo-entrenamiento, de tal manera que el proceso de entrenamiento se realiza mientras la mejoría del rendimiento en la base de entrenamiento no es superada por la mejoría en la base de validación (Kuncheva, 2005).

La mejor forma de organizar el experimento en entrenamiento y prueba realmente depende de las características de la base de datos. En grandes volúmenes de datos un *10-fold cross-validación*, que es uno de los más usados, no es muy útil pues demoraría demasiado la validación, aquí quizás un *método H* sería mejor. Al igual que una base de datos pequeña no permitiría el uso del *método H* ya que la base de entrenamiento sería demasiado pequeña y divisiones de la misma pueden dejar una base no completa, quizás aquí el método *boostrap* sea mejor (Kohavi., 1995).

3.2 Descripción de los casos de estudio.

Anteriormente se trató el comentario de la base de casos a estudiar, la cual consistía en una base de imágenes perteneciente a los registros mensuales de las pizarras de metro-contadores de las centrales analógicas que ETECSA tiene instaladas en todos los municipios del país. Debido a la falta de estándares en la captura de las imágenes teniendo en cuenta aspectos tales como: luminosidad, ángulo de enfoque, encuadramiento y posición de dispositivos de seguridad no es posible lograr una buena segmentación y legibilidad de la imagen; y es por ello que hasta el momento se consideró fijar dimensiones a los fragmentos de imágenes a tratar que contenían la región de interés. Otra de las dificultades que presentan tales imágenes como resultado de lo anteriormente expuesto es la presencia de ruido en la generalidad del conjunto. No obstante es posible formular 2 variantes para probar la clasificación.

Caso 1: Entrenar partir de la división del conjunto muestral en un conjunto de entrenamiento (CE) y un conjunto de prueba (CP) como se hace normalmente. Para entrenar la red nos basamos entonces en la selección de una muestra de los patrones de

los dígitos de las imágenes los cuales de modo general presentan diversas señales de ruidos.

Caso 2: Entrenar la red basándonos en la generación de los patrones de los dígitos perfectamente concebidos.

El propósito de nuestra red radica entonces en eliminar todo el ruido posible que aparezca de forma irregular para lograr que sea lo mas semejante posible a aquellos que fueron entrenados.

3.2.1 Validación de los resultados.

Caso de estudio 1:

Se tomo de varias imágenes 10 muestras de cada dígito los cuales estaban caracterizados por zonas ruidosas en la parte superior e inferior de la matriz binaria representativa y una menor frecuencia en la parte central. Se decidió hacer una prueba *10-fold cross-validación* para el conjunto muestral de los 100 representantes de todos los digitos dividiendo en CE y CP en una razón de 80% y 20% arbitrariamente en cada una de las 10 pruebas.

Clases	1	2	3	4	5	6	7	8	9	0	%
1	5	0	1	8	0	2	3	0	1	0	25
2	0	5	0	5	0	0	9	0	1	0	25
3	0	1	5	4	0	4	0	0	5	1	25
4	5	2	4	5	0	0	0	1	3	0	25
5	0	0	1	0	0	6	0	8	5	0	0
6	0	0	7	0	4	2	1	2	4	0	10
7	0	5	0	3	0	0	12	0	0	0	60
8	1	0	3	0	0	2	0	8	6	0	40
9	4	0	5	0	0	3	0	1	6	1	30
0	0	0	2	0	2	0	0	2	10	4	20

%	75	65	140	125	30	95	125	110	205	30	31

Figura 3.2 Matriz de confusión del caso 1.

La última columna indica el por ciento de clasificación correcta por cada clase y la última fila el por ciento de sobreclasificación por clase tomándose el 100% como medida estándar. La intersección de ambas corresponde a la clasificación general.

Es apreciable según la anterior matriz de confusión el bajo por ciento de clasificaciones correctas en todas las clases donde no se supera el 60% e incluso existen dificultades en varias clases en las cuales quedan por debajo del 30%. En la última fila también es observable que los patrones representantes del dígito 9 fueron seleccionados como solución en una magnitud enorme lo cual indica que este presenta las mejores características en la distribución del nivel de grises de sus píxeles respecto a las demás clases para la muestra de control analizada. Finalmente de los 200 resultados obtenidos solo 62 fueron correctos de todas las clases para un muy bajo 31% de clasificación general.

Caso de estudio 2:

Fueron seleccionadas 25 imágenes las cuales contienen 6 dígitos cada una para un total de 150 reconocimientos. Para ello se calculo la matriz de confusión.

Clases	1	2	3	4	5	6	7	8	9	0	%
1	7	0	0	3	0	0	3	0	0	3	43.7
2	2	7	0	4	0	0	3	0	0	0	43.7
3	8	0	3	3	0	0	1	0	0	0	20
4	2	0	0	8	0	0	4	1	0	0	53.3
5	2	0	0	3	3	3	1	0	0	0	25
6	2	0	0	3	0	6	0	0	0	0	54.5
7	2	2	0	2	0	0	4	0	0	0	40
8	3	0	0	1	0	3	1	4	2	0	28.5

9	2	0	0	1	0	0	1	0	11	0	73.3
0	11	0	0	3	0	1	1	0	8	2	7
%	41	9	3	31	3	13	19	5	21	5	36.6

Figura 3.3 Matriz de confusión del caso 2.

La última columna indica el por ciento de clasificación correcta por cada clase y la última fila el número de resultados dados con esos dígitos.

En este caso se alcanza superar por 5% la clasificación correcta general siendo un poco mayores los por cientos de clasificación por clases, y al menos todas con alguna representación.

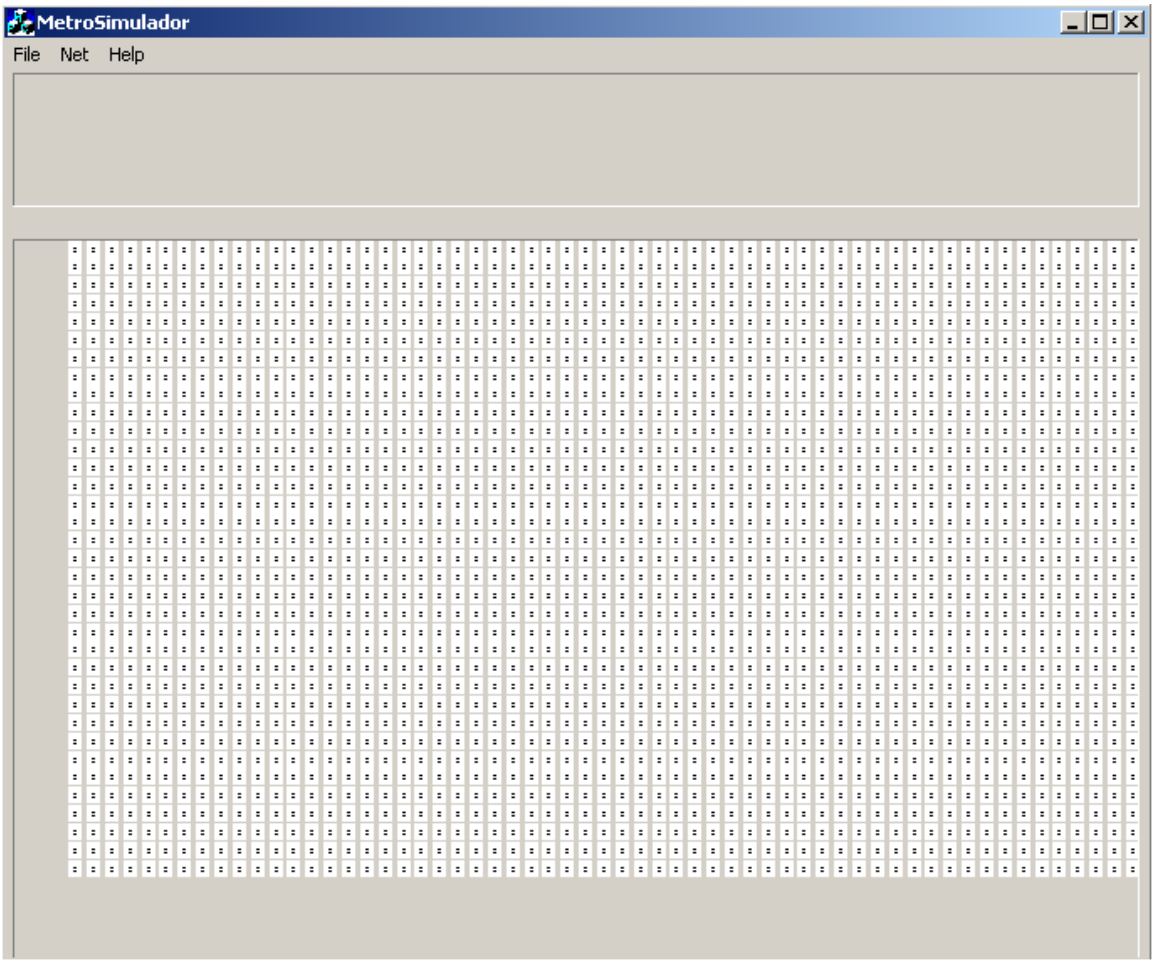
3.3 Manual de Usuario.

Requerimientos del software

- Sistema Operativo Windows 2000, o versiones superiores.
- Fichero MetroSimulador.exe.
- Fichero mlStuffsPk.dll ubicado en el mismo directorio que el ejecutable.

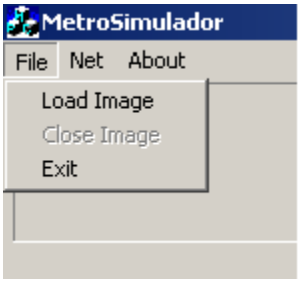
Pasos a seguir para la interacción con el software.

1. Ejecución del fichero MetroSimulador.exe.



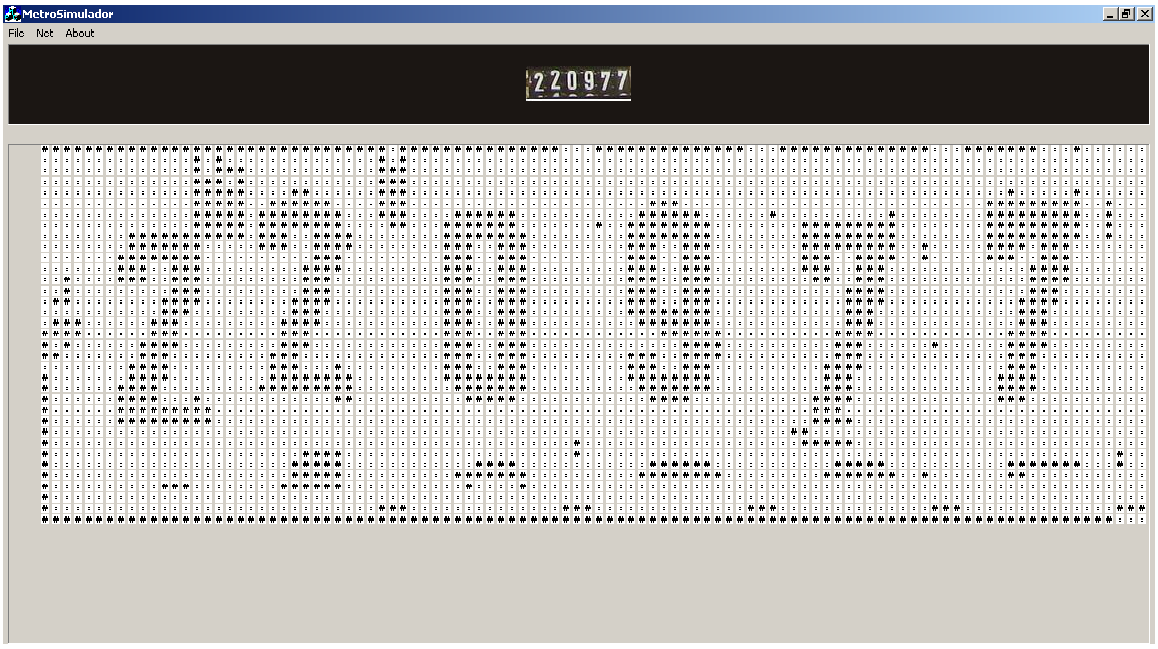
Como puede observarse la aplicación consta de 3 menús: File, Net y Help.

En **File** se encapsulan las opciones relativas al fichero imagen y el cierre de la aplicación. **Net** contiene todo lo referente al funcionamiento de la red neuronal desde su inicialización hasta el almacenaje de los resultados. **Help** solamente tiene una breve información acerca de la aplicación.



Las opciones para **File**:

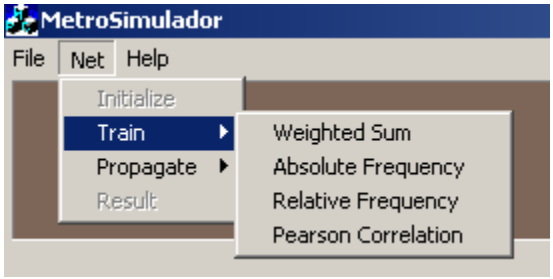
- *Load Image*: Abre el diálogo de Windows de apertura de documentos para que una imagen de formato JPG sea seleccionada. Esta opción activa *Close Image.*
- *Close Image:* Cierra la imagen cargada y limpia la matriz binaria de la interfaz. Aparece inicialmente desactivada puesto que no se puede cerrar una imagen si no ha sido cargada. Solo se activa cuando hay una imagen cargada.
- *Exit:* Cierre de la aplicación.



He aquí la representación de la imagen junto a su matriz binaria de pixeles luego de cargar una imagen.

Las opciones para **Net**:

- *Initialize*: Aparece inicialmente desactivada hasta que una imagen sea cargada. Su función es inicializar la red o sea preparar todas las variables que van a ser utilizadas en el procesamiento (Pone sus valores a 0).
- *Train*: Despliega otro conjunto de opciones relativas a los distintos métodos de entrenamiento que pueden ser usados los cuales se activan una vez que la red haya sido inicializada.



  - *Weighted Sum*: Entrena la red usando la variante del calculo de pesos por la suma pesada.
  - *Absolute Frequency*: Entrena la red usando la variante del calculo de pesos por la frecuencia absoluta.
  - *Relative Frequency*: Entrena la red usando la variante del calculo de pesos por la frecuencia relativa.
  - *Pearson Correlation*: Entrena la red usando la variante del calculo de pesos por la correlación de Pearson.
- *Propagate*: Despliega otro conjunto de opciones relativas a los distintos métodos para el calculo de las activaciones de las neuronas de la red los cuales pueden ser usados una vez entrenada la red.

- ➢ *Sequential*: Calcula las activaciones de las neuronas de la red de forma secuencial.
- ➢ *Parallel*: Calcula las activaciones de las neuronas de la red de forma paralela.

- *Result*: Abre un diálogo de salva para crear un fichero texto y salvar los resultados del reconocimiento, o sea el número que reconoció.

Las opciones para **Help**:

- *About*: Muestra información relativa a la creación del software.

Conclusiones Parciales.

La validación se realiza utilizando un método *hold-out* en el caso de estudio 1. La muestra se divide en 80% para entrenar y 20% para explorar el funcionamiento de la red. Los resultados muestran un % de clasificación por debajo del 60% por clase para un 31% de clasificación general. Para el segundo caso se obtuvo una clasificación correcta de 36.6% superando la anterior.

La interfaz usuario obtenida es amigable y da respuesta a los requerimientos del cliente final, sin que este tenga que conocer los principios del funcionamiento de una red neuronal.

Conclusiones Generales

Se desarrolló un OCR que permita automatizar el proceso de lectura de los metros contadores para el cálculo del importe telefónico a facturar por cliente, a partir de fotos digitales de las centrales analógicas de ETECSA, mediante el uso de técnicas de procesamiento de imágenes y redes neuronales. El OCR implementado presenta las siguientes características:

- Se implementa como método de segmentación el Método Basado en Píxeles porque sólo se necesita el valor de gris de un pixel y fijar umbrales para estos para así identificar las objetos de interés del resto pasando por alto el evitar cambiar el tamaño del objeto. Los métodos utilizados para la descompresión son la codificación Huffman y RLE, atendiendo a sus bondades con los formatos JPEG.
- Se diseñó e implementó un modelo de Holpfield Discreto. La implementación incluye cuatro variantes de calcular los pesos de la RNA y se incluyeron las posibilidades de funcionamiento secuencial asíncrono y funcionamiento paralelo síncrono.
- La implementación del OCR es sencilla y asequible, fácil de modificar de acuerdo al diseño de clases concebido, el software da respuesta a los requerimientos del cliente final.
- El reconocedor desarrollado fue validado utilizando un método *hold-out* en el caso de estudio 1. La muestra se divide en 80% para entrenar y 20% para explorar el funcionamiento de la red. Los resultados muestran una clasificación correcta bastante baja de 31%. Para el segundo caso se obtuvo una clasificación correcta de 36.6%.

Recomendaciones

Para el buen funcionamiento de la aplicación se hace necesario que la empresa fije estándares respecto a la captura de las imágenes teniendo en cuenta aspectos tales como: luminosidad, ángulo de enfoque, encuadramiento y posición de dispositivos de seguridad lo cual contribuirá a lograr una buena segmentación y legibilidad de la imagen.

La implementación de otras funciones de similitud que asignen pesos de importancia a los pixeles de acuerdo a su posición dentro de la matriz binaria permitiría una mejor toma de decisiones en la identificación de un patrón.

Bibliografía

BETANCOURT, G. A. (2005) Las Maquinas de Soporte Vectorial. *Scientia et Technica Año XI, No 27, Abril 2005. UTP*.

BONET, I. (2001) Implementación de un OCR para incorporarlo a los softwares de gestión de oficinas distribuidos por Avilasoft. *Bioinformática.* Santa Clara, UCLV Martha Abreu.

BONET, I., DÍAZ, A., BELLO, R. & SARDIÑA, Y. (2002) Learning optimization in a MLP Neural Network Applied to OCR. *MICAI 2002: Advances in Artificial Intelligence. LNAI,* 2313**,** 292-300.

BURGES, C. J. C. (1998) *"A Tutorial on Support Vector Machines for Pattern Recognition"*.

CLEARY, J. G. A. W., I.H. (1984) "Data Compression using adaptive coding and partial sting matching". *IEEE Transactions on Communications,* Vol 32**,** 396-402.

CROSSLAND, M. D., WYNNE, B.E. & PERKINS, W.C. (1995) Spatial decision support systems: An overview of technology and a test of efficacy. *Decision Support Systems.*

DIAZ SARDIÑAS, A. (2000) "Heurísticas para el diseño de redes neuronales tipo Multilayer Perceptron (MLP) que utilizan Backpropagation (BP) como algoritmo de aprendizaje". . *Inteligencia Artificial.* Santa Clara, Martha Abreu.

FOWLER, J. A. Y., R. (1994) "Lossless Compression of Volume Data". *Symposium on Volume Visualization***,** pp. 43-50.

FREEMAN, J. A. S., DAVID M. (1993) "Redes neuronales: algoritmos, aplicaciones y técnicas de programación.". *Addison Wesley Iberoamericana*.

GARCÍA, M. M. E. A. (1996) A model and its different applications to case-based reasoning. *Knowledge-based systems*

HENST, C. V. D. (1997) Maestros del Web. España.

HILERA GONZÁLEZ, J. R. M. H., VÍCTOR JOSÉ. (1995) Redes neuronales artificiales: fundamentos, modelos y aplicaciones. *Addison-Wesley Iberoamericana*.

HOPFIELD, J. J. (1982) "Neural networks and physical systems with emergent properties". *Proceedingns of the National Academy of Sciences.*

HUFFMAN, D. (1952) "A Method for the Construction of Minimum Redundancy Codes". *Proceedings of de IRE 40(9):*.

KOHAVI., R. (1995) "A study of cross-validation and bootstrap for accuracy estimation and model selection.". *International Joint Conference on Artificial Intelligence.*

KUNCHEVA, L. I. (2005) Combining Pattern Classifiers, Methods and Algorithms. New York, Wiley Interscience.

MATICH, D. J. (2001) Redes neuronales. Conceptos básicos y aplicaciones. .

MORENO, J. J. M. (2002) Redes Neuronales Artificiales aplicadas al analisis de datos. Palma de Mayorca, España.

NAMANE, A. A., M. GUESSOUM, A. SOUBARI, EL HOUSSINE. MEYRUEIS, P. BRUYNOOGHE M. (2005) Sequential neural network combination for degraded machine printed character recognition. *SPIE proceedings series,* 5676**,** pp. 101-110.

PLAZA, A. A. A. E. (1994) *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches.*, IOS Press.

PROVOST, F. F., T. (1997) Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions.

QUINTERO MARCELA, A. E. (2006) Guía para el uso de Árboles de Decisión. Alternativas de Uso de la Tierra para los Llanos Orientales de Colombia.

RAITA, T. T., J. (1987) "Predictive Text Compression by Hashing". *ACM Conference on Information Retrieval.*

RUMELHART, D. E., HINTON, A. G. E. & WILLIAMS, A. R. J. (1986) Learning internal representations by error propagation MIT Press.

RUSS, J. C. (1991) The Image Processing Handbook.

SALOMON, D. (2002) A Guide to Data Compresión Methods. New York, Springer-Verlag.

VAPNIK, V. (1995) *The Nature of Statistical Learning Theory*, New York, Springer-Verlag.

VÁZQUEZ, R. V. (2004) Implementación y prueba del estándar JPEG 2000. *Electronica y Sistemas.* Coruña, Universidad de Coruña.

WALLACE, G. K. (1991) "The JPEG Still Picture Compression Standard.". IN ELECTRONICS, I. I. T. O. C. (Ed.). Massachusetts.

WELCH, T. A. (1984) "A Technique for High Performance Data Compression". *Computer,* Vol 17.

WITTEN, I. H., NEAL, R.M., CLEARY, J.G. (1987) "Arithmetic Coding for Data Compression". *Communication of the ACM,* Vol 30**,** pp. 520-540.

WITTEN. I.H., M., A. AND BELL, T.C. (1999) *"Managing Gigabytes: Compressing and Indexing Documents and Images",* San Francisco, Academic Press.

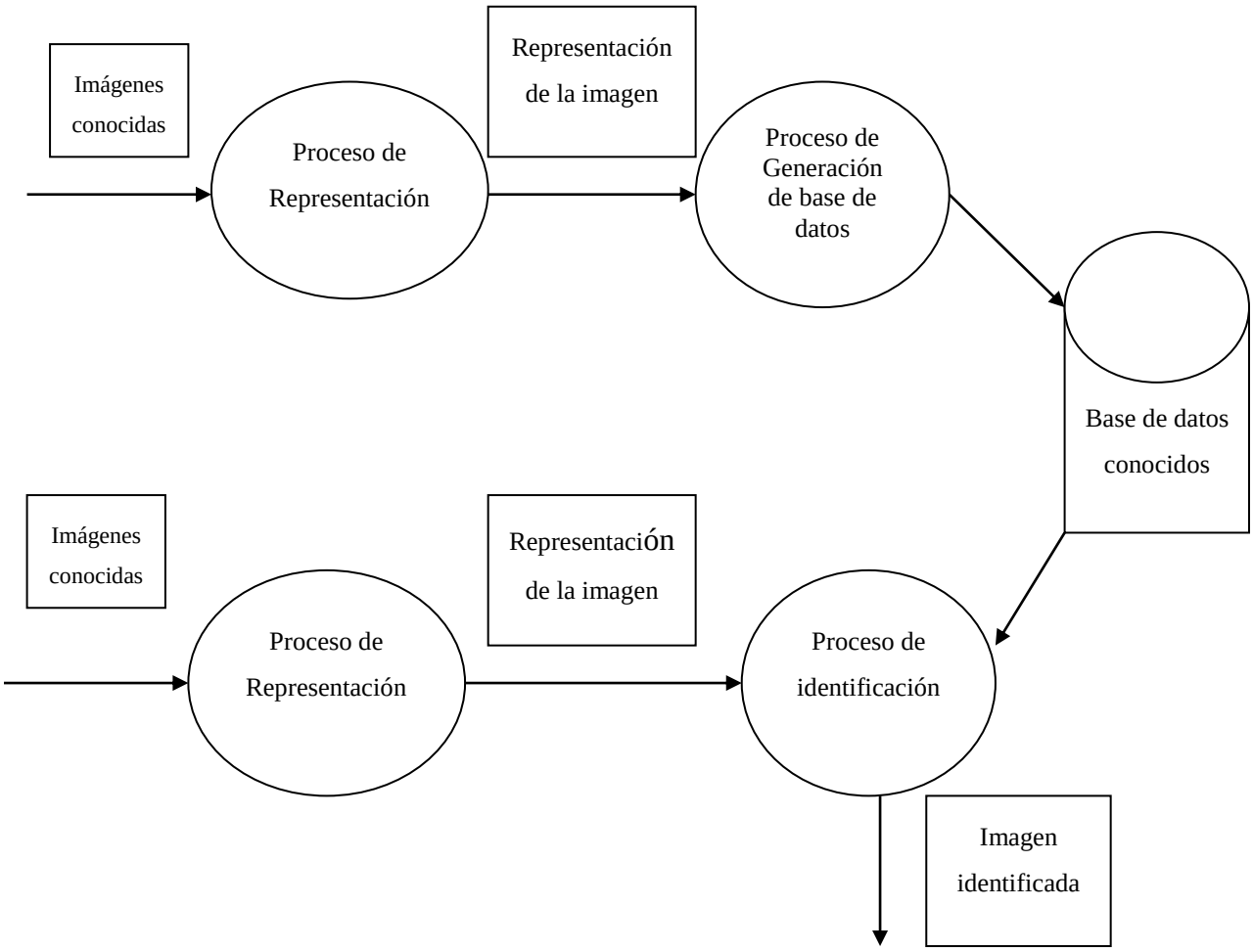
YSIQUIO, A. N. F. (2004) Modelado de Sistemas de control de un robot manipulador basado en procesamiento digital de Imágenes. *Departamento de Ingeniería en Sistemas Computacionales.* Puebla, Mexico, Universidad de las Américas Puebla.

ZIV, J. A. L., A. (1977) "A Universal Algorithm for Secuential Data Compression". *IEEE Trans. Information Theory,* vol. 23**,** pp. 337-343.

ZIV, J. A. L., A. (1978) "Compression of Individual Secuences via Variable-Rate Coding". *IEEE Transactions on Information Theory,* vol. 24**,** pp. 530-536.

Anexos

Anexo 1: Esquema de un OCR.



Anexo 2: Formato del fichero JPG.

Este formato está compuesto por markers. Cada marker está prefijado por FF (en hexadecimal). En su comienzo cada marker tiene dos byte que indican la longitud de la información dentro de él, incluyendo estos dos bytes. La imagen comienza con un marker que tiene 2 byte indicando el comienzo y cuyo valor debe ser FFD8.

Después pueden seguir los siguientes markers:

Marker de comienzo de la imagen (FFD8):

Marker JFIF (FFE0):

Campo	**Descripción**	**Longitud**
Longitud	Longitud del marker.	2 bytes.
Identificador	Debe ser 4A, 46, 49, 46, 00 (el código ASCII de al cadena "JFIF" terminada en cero).	5 bytes.
Versión	Versión.	2 bytes.
Unidades	Tipos de unidades para las densidades de X y Y	1 byte.
DensidadX	Densidad de X.	2 bytes.
DensidadY	Densidad de Y.	2 bytes.
Xthumbnail	0 si X es thumbnail.	1 byte.
Ythumbnail	0 si Y es thumbnail.	1 byte.
(RGB)n	Valores RGB para pixels thumbnail. n= Xthumbnail* Ythumbnail	3n bytes.

Marker de la tabla de Cuantización (FFDB):

Campo	**Descripción**	**Longitud**
Longitud	Tamaño del marker.	2 bytes
Indice	Índice de la tabla.	1 byte.

Elementos	El resto son los 64 elementos de la tabla.	64 byte.

Marker de la Tabla de Huffman (FFC4):

Campo	**Descripción**	**Longitud**
Longitud	Tamaño del marker.	2 bytes.
Indice	Si es mayor que 15 es una tabla AC sino es DC.	1 byte.
Bits	Los 16 bits de la tabla.	16 bytes.
Valores	Los valores de la tabla.	Suma de los 16 bits.

Marker de Comienzo del Frame (FFCO):

Campo	**Descripción**	**Longitud**
Longitud	Tamaño del marker.	2 bytes.
Precisión	Precisión en bits, usualmente 8.	1 byte.
X		2 byte.
Y		2 bytes.
Nf	Número de componentes de color en la imagen.	1 byte.
Repetir tantas veces como components haya:		
ID Comp	Identificador de la componente.	1 byte.
HV	Factores sampling. H está enlos primeros 4 bits y V en los últimos. Indican el tamaño de la componente a la que están asociados.	1 byte.
NumCuant	Número de la tabla de cuantización.	1 byte.

Marker del Comienzo del Scan (FFDA):

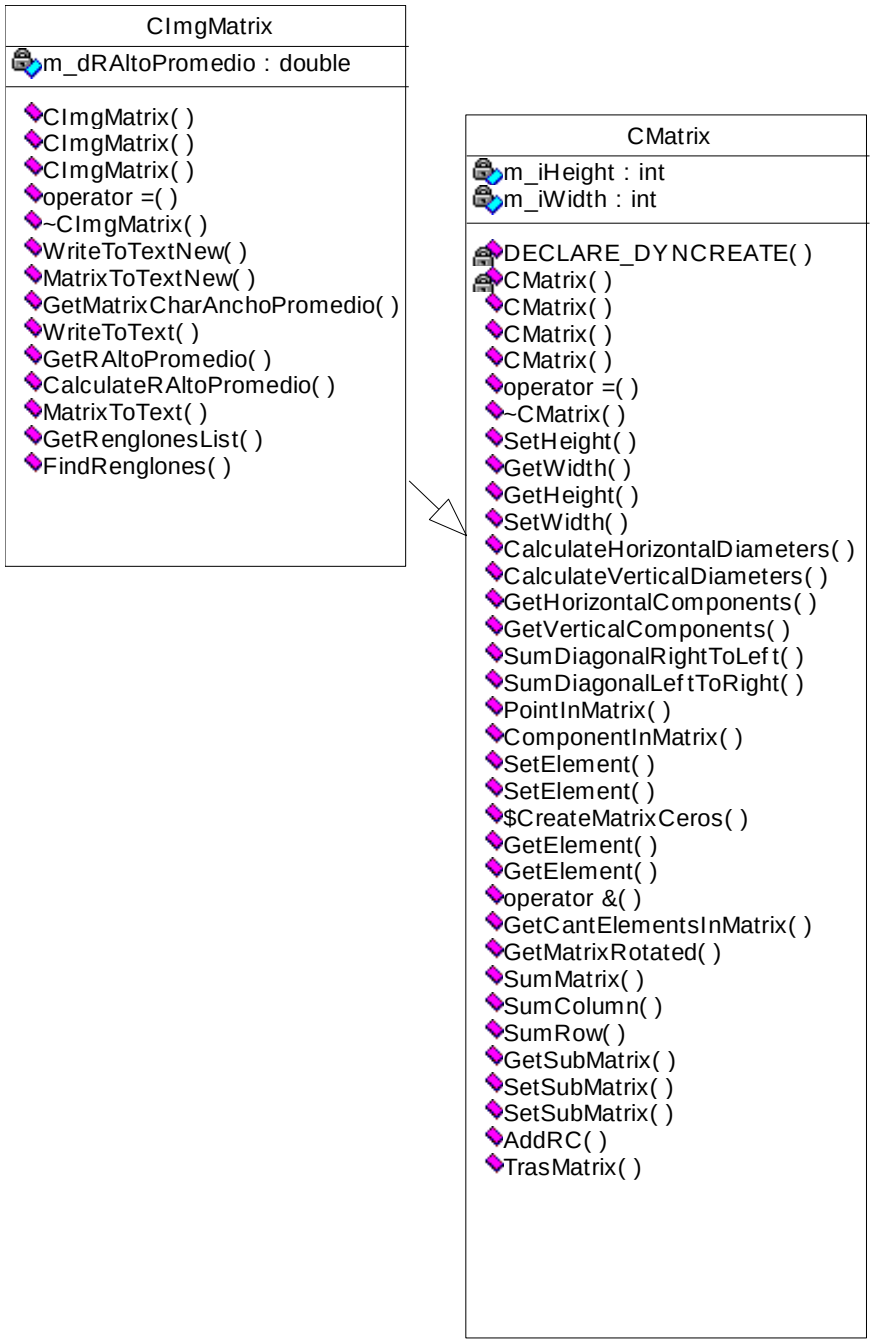
Campo	**Descripción**	**Longitud**
Longitud	Tamaño del marker.	2 bytes.
N	Númerode components en este Scan.	1 byte.
N veces:		
ID	Indice de la componente	1 byte.
Cant AC-DC	Los primeros 4 bits tienen el número de la tabla AC y los otros 4 el de la DC.	1 byte.

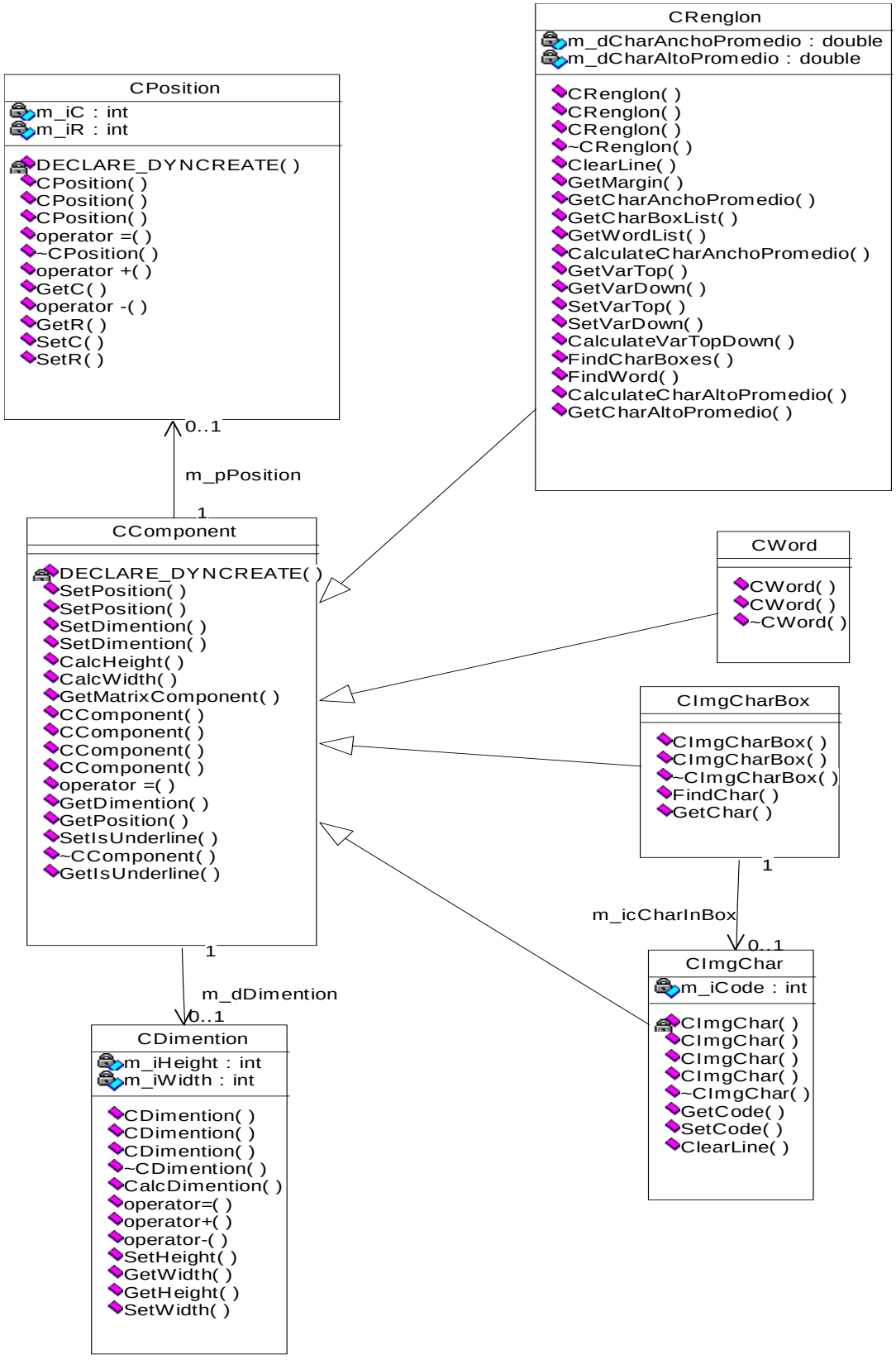
Marker de Comentarios (FFFE):

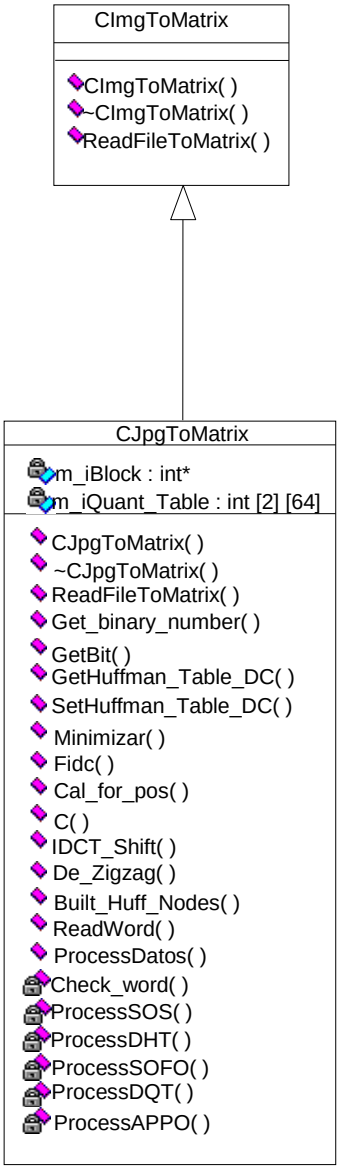
Campo	**Descripción**	**Longitud**
Longitud	Tamaño del marker.	2 bytes.
Lo que se quiera.		

Marker de Fin de Imagen (FFD9).

Anexo 3: Diagrama de la relación de algunas de las clases más importantes.




CRenglon
m_dCharAnchoPromedio : double
m_dCharAltoPromedio : double
CRenglon( )
CRenglon( )
CRenglon( )
~CRenglon( )
ClearLine( )
GetMargin( )
GetCharAnchoPromedio( )
GetCharBoxList( )
GetWordList( )
CalculateCharAnchoPromedio( )
GetVarTop( )
GetVarDown( )
SetVarTop( )
SetVarDown( )
CalculateVarTopDown( )
FindCharBoxes( )
FindWord( )
CalculateCharAltoPromedio( )
GetCharAltoPromedio( )
CPosition
m_iC : int
m_iR : int
DECLARE_DYNCREATE( )
CPosition( )
CPosition( )
CPosition( )
operator =( )
~CPosition( )
operator +( )
GetC( )
operator -( )
GetR( )
SetC( )
SetR( )
0..1
m_pPosition
1
CComponent
DECLARE_DYNCREATE( )
SetPosition( )
SetPosition( )
SetDimention( )
SetDimention( )
CalcHeight( )
CalcWidth( )
GetMatrixComponent( )
CComponent( )
CComponent( )
CComponent( )
CComponent( )
operator =( )
GetDimention( )
GetPosition( )
SetIsUnderline( )
~CComponent( )
GetIsUnderline( )
CWord
CWord( )
CWord( )
~CWord( )
CImgCharBox
CImgCharBox( )
CImgCharBox( )
~CImgCharBox( )
FindChar( )
GetChar( )
1
m_icCharInBox
0..1
CImgChar
m_iCode : int
CImgChar( )
CImgChar( )
CImgChar( )
CImgChar( )
~CImgChar( )
GetCode( )
SetCode( )
ClearLine( )
1
m_dDimention
0..1
CDimention
m_iHeight : int
m_iWidth : int
CDimention( )
CDimention( )
CDimention( )
~CDimention( )
CalcDimention( )
operator=( )
operator+( )
operator-( )
SetHeight( )
GetWidth( )
GetHeight( )
SetWidth( )


CImgToMatrix
CImgToMatrix( )
~CImgToMatrix( )
ReadFileToMatrix( )
CJpgToMatrix
m_iBlock : int*
m_iQuant_Table : int [2] [64]
CJpgToMatrix( )
~CJpgToMatrix( )
ReadFileToMatrix( )
Get_binary_number( )
GetBit( )
GetHuffman_Table_DC( )
SetHuffman_Table_DC( )
Minimizar( )
Fidc( )
Cal_for_pos( )
C( )
IDCT_Shift( )
De_Zigzag( )
Built_Huff_Nodes( )
ReadWord( )
ProcessDatos( )
Check_word( )
ProcessSOS( )
ProcessDHT( )
ProcessSOFO( )
ProcessDQT( )
ProcessAPPO( )

Anexo 4: Medida de Distancias.

Distancia Euclidiana:	$D(X,Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$
Distancia de Manhattan:	$D(X,Y) = \sum_{i=1}^{N} \lvert x_i - y_i \rvert$
Distancia de Chebychev:	$D(X,Y) = \max_{i=1}^{N} \lvert x_i - y_i \rvert$
Distancia de Hamming:	$D(X,Y) = \sum_{i=1}^{N} d(x_i, y_i)$ $d(x_i, y_i) = \begin{cases} 1 & si \quad x_i \neq y_i \\ 0 & si \quad x_i = y_i \end{cases}$