3 Diseño del Sistema de Reconocimiento de Letras.

3.1. Introducción.

El sistema fue desarrollado bajo el sistema operativo Windows XP de Microsoft, utilizando el entorno de desarrollo C++Builder Profesional versión 6 de Borland Inc.

Se tiene una colección de imágenes en formato GIF que han sido utilizados por [Linares & Spínola 00] y [Navarrete 02]. Estas servirán como entrada para el sistema.

Cabe mencionar que primero se realizaron unos programas para entender el funcionamiento del clasificador del Vecino Más Cercano, en primera para 2 dimensiones (ver figura 3.1) posteriormente se generalizó para N-Dimensiones (ver figura 3.2).

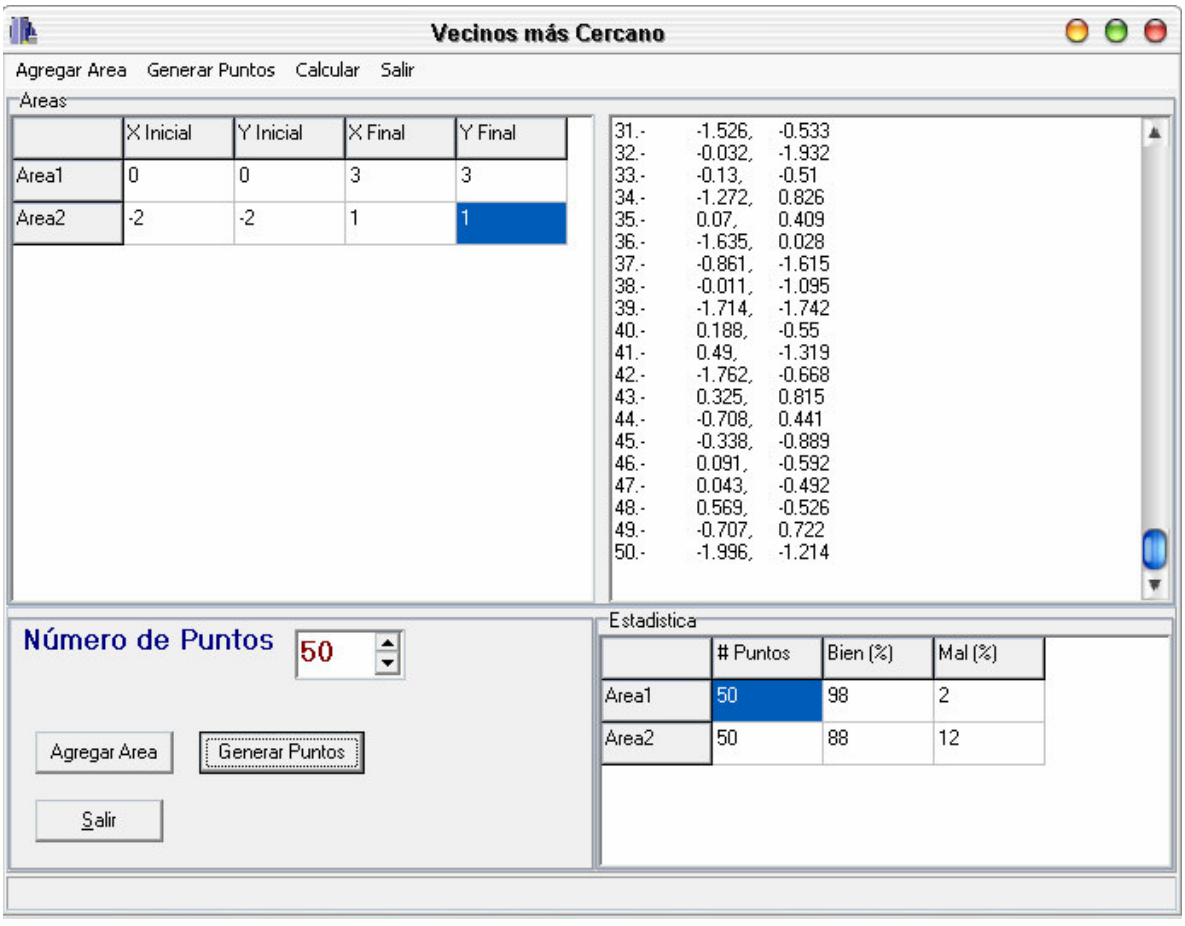


Figura 3.1 Corrida del programa Vecino Más Cercano en 2-D.

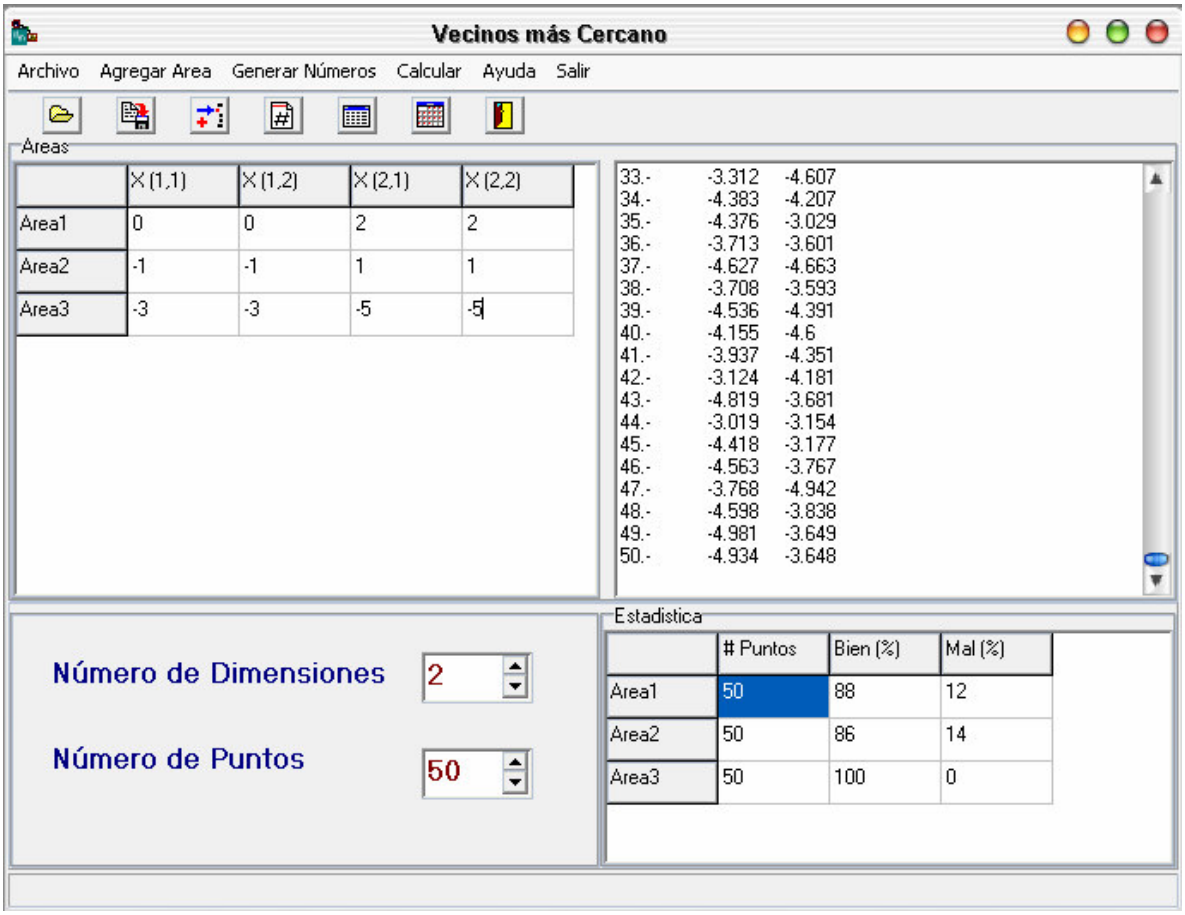


Figura 3.2 Corrida del programa Vecino Más Cercano en N-D.

Para el programa de N-Dimensiones se calculó el representante (centroide) de la clase, para esto se calcula su media ponderada por cada coordenada, usando la ecuación 2.7, esto se hizo para ahorra tiempo de procesamiento.

Para probar que el algoritmo funcionara correctamente se utilizó un ejemplo clásico: Clasificar números generados al azar en N posibles áreas. El programa de prueba funciona de la siguiente manera, primero se tienen que definir áreas donde se van a generar los números, enseguida generar aleatoriamente número reales entre esas áreas. Posteriormente se pueden calcular a qué clase pertenecen usando como entrada los números anteriores o

generar nuevos números. El programa de prueba visualiza una tabla donde muestra por cada área el número de puntos, cuantos aciertos y fallos, es un aprendizaje supervisado.

El programa lee desde un archivo de texto las áreas, para esto el archivo debe ir de la siguiente forma:

Número de Dimensiones

Número de Áreas

x(1,1,..), x(1,2,...),....,x(1,n,...)

x(1,1,..), x(1,2,...),....,x(1,n,...)

x(1,1,..), x(1,2,...),....,x(1,n,...)

x(1,1,..), x(1,2,...),....,x(1,n,...)

} Puntos que definen cada área, donde se van a generar los números.

Ejemplo

3 (3 características o dimensiones de los puntos)

3 (3 posibles áreas)

1, 2.2, 3, 4, 5, 6, 7, 8, 9

11, 12.2, 13.5, 14, 15, 16, 17, 18, 19

21, 22.2, 23.5, 24, 25, 26, 27, 28, 29

} Puntos de cada área

Este ejemplo y varios más funcionaron correctamente, siendo clasificados los puntos con un buen porcentaje. La tabla 3.1 presenta los resultados obtenidos al clasificar 50 puntos de cada área de las 4 áreas, La figura 3.3. Es la grafica de los puntos generados en las 4 áreas definidas.

	# de Puntos	Bien (%)	Mal (%)
Area1	50	100	0
Area2	50	96	4
Area3	50	96	4
Area4	50	100	0

Tabla 3.1 Porcentaje de reconocimiento del programa Vecino Más Cercano N-Dimensión.

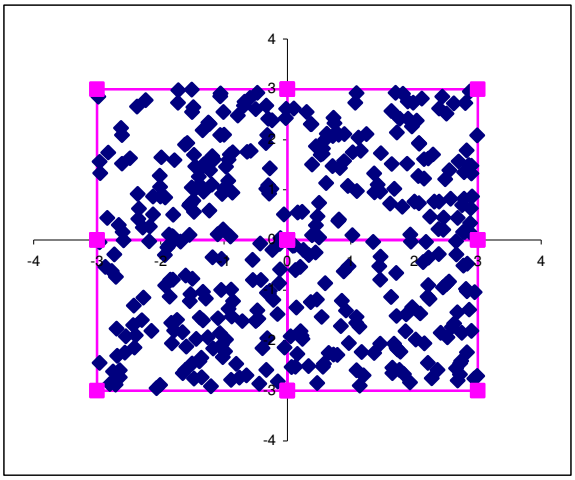
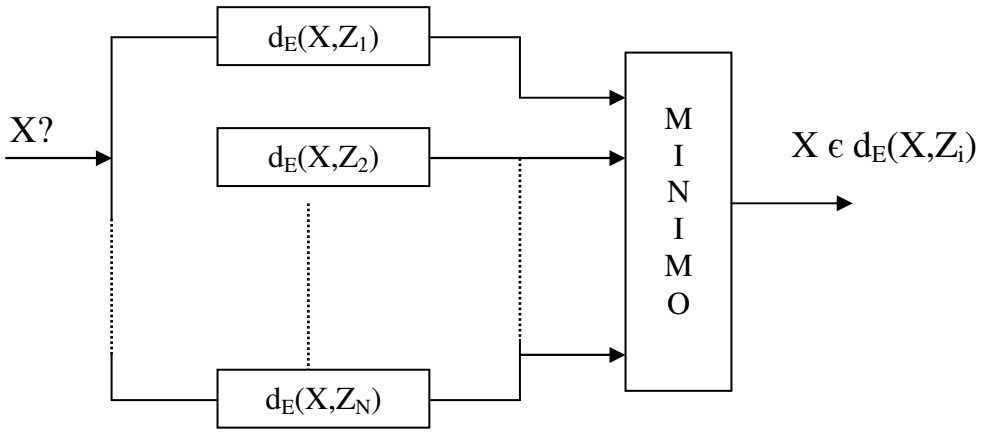


Figura 3.3 Grafica en 2 Dimensiones con 4 regiones.

3.1.1 Análisis del Clasificador por Medio del Vecino Más Cercano.

El siguiente diagrama de bloques ilustra el proceso de clasificación por medio de la distancia euclidiana.



Traducido al siguiente diagrama de flujo

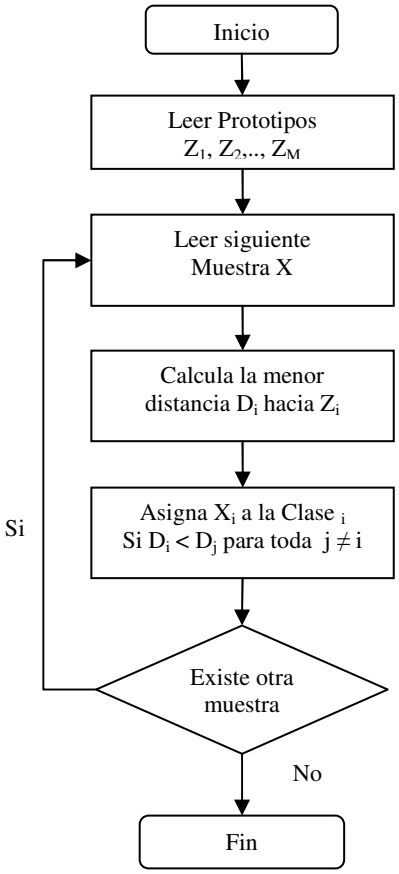


Figura 3.4 Diagrama de Flujo de un Reconocedor por la Distancia Euclidiana.

Posteriormente se desarrolló un programa para el cálculo de *clusters* usando el algoritmo de K-*Means* (ver figura 3.5) descrito en la sección 2.3.3. El programa funciona de la siguiente manera.

Primero se tiene que abrir un archivo de patrones (ver figura 3.6) y posteriormente ejecutar el proceso de calcular centroides. Aquí el programa requiere que el usuario proporcione el número de clases (K) que se crearan. Una vez calculado los centroides zi y los conjuntos Si, el programa visualizara los resultados (ver figura 3.7) para que posteriormente el usuario guarde en un archivo de texto estos resultado.

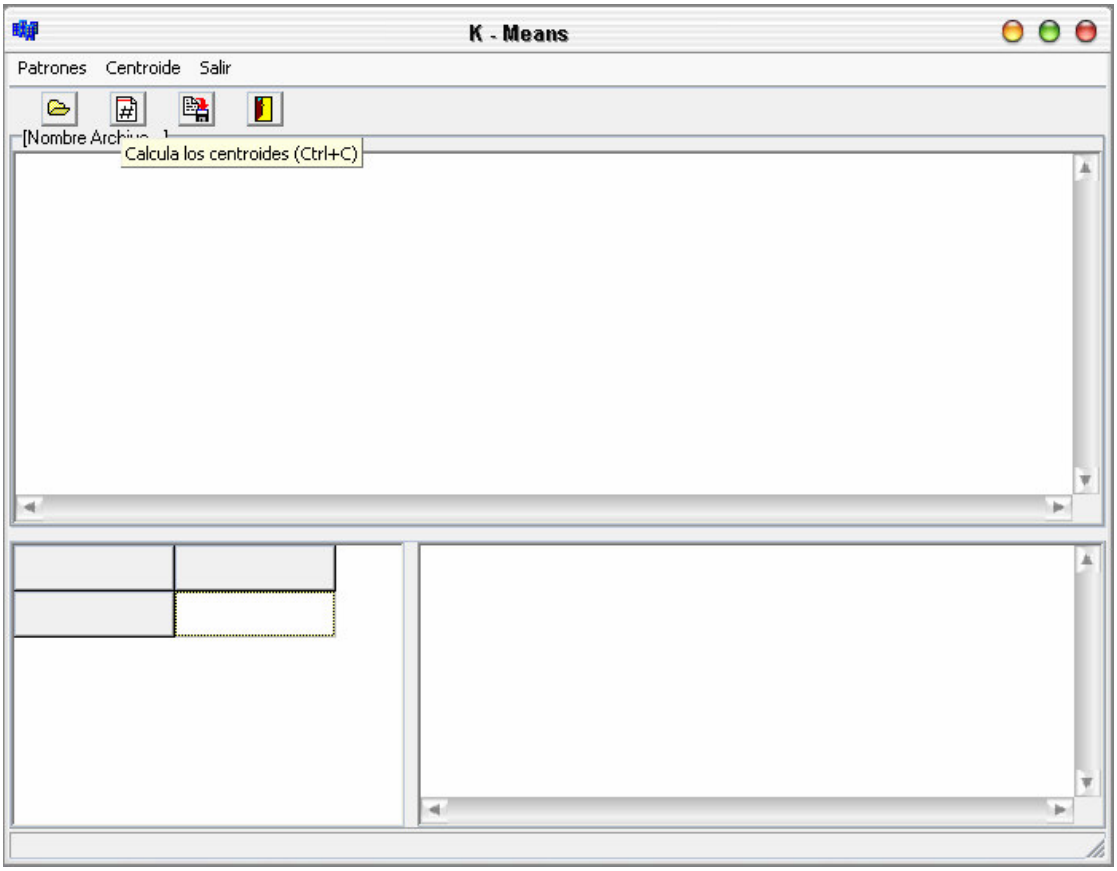


Figura 3.5 Programa K-Means.

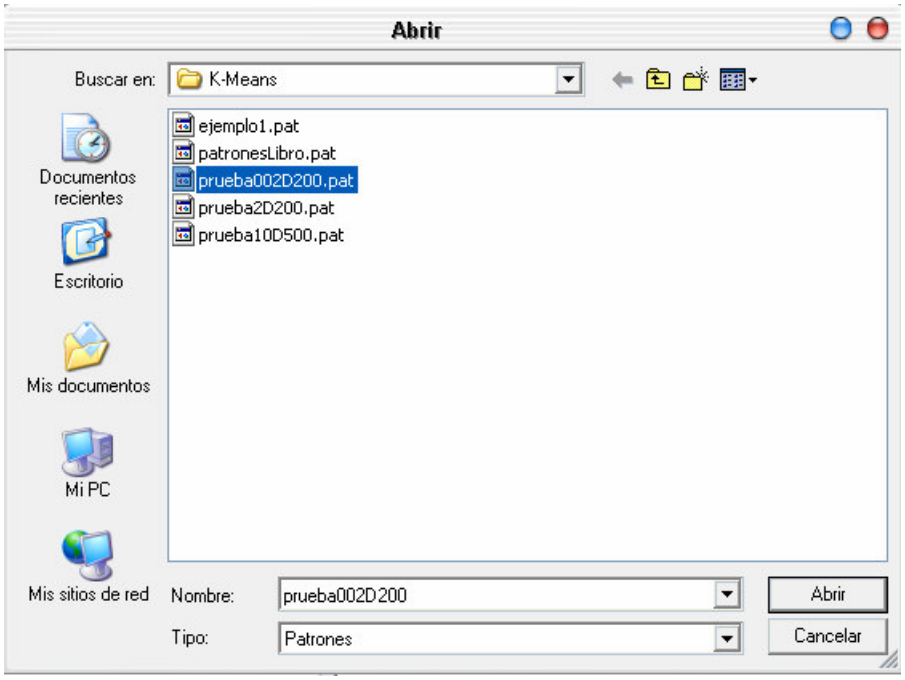


Figura 3.6 Ventana para abrir un archivo de patrones.

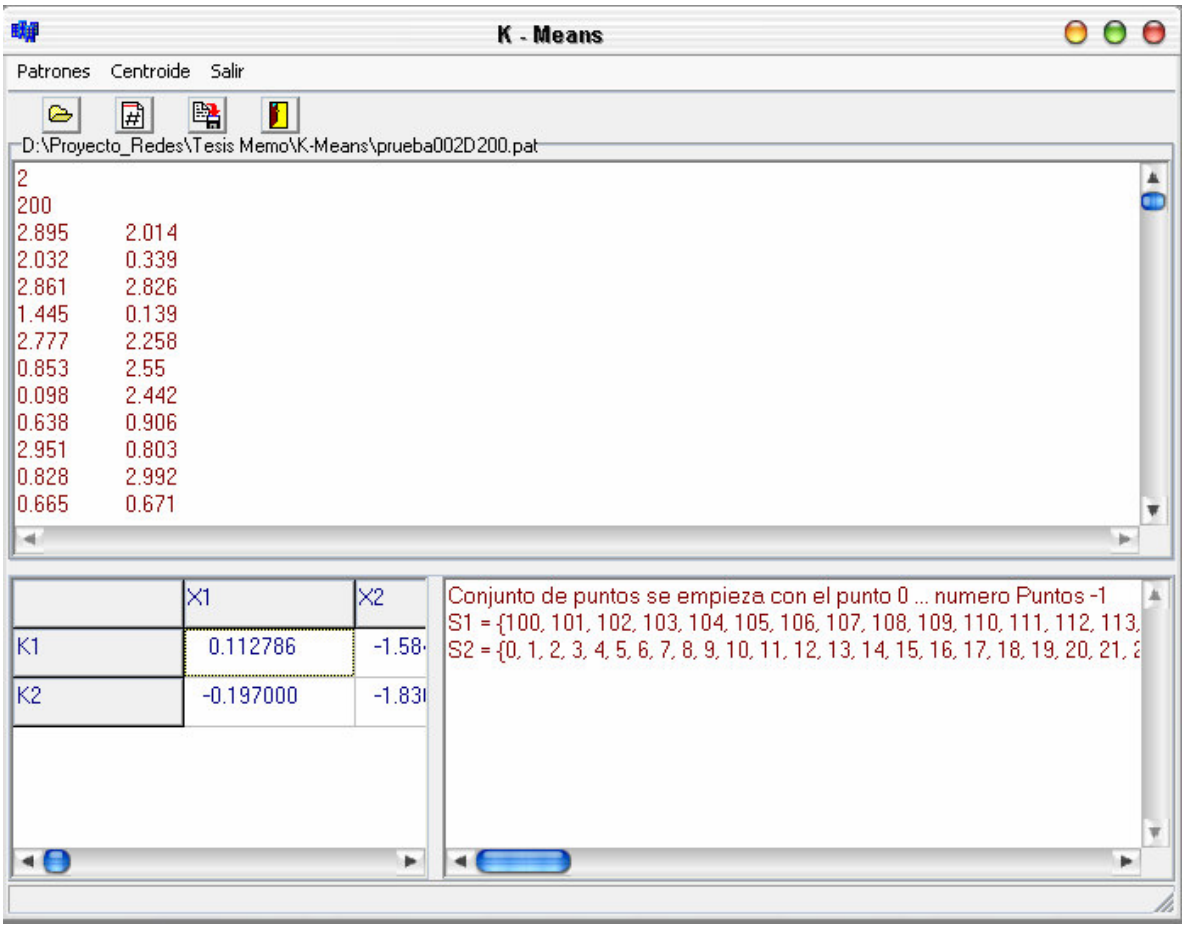


Figura 3.7 Programa K-Means visualizando los resultados.

Para el archivo de patrones se tiene que seguir el siguiente formato.

Número de dimensiones

Número de puntos

coordenada1 coordenada2 coordenadaN

coordenada1 coordenada2 coordenadaN

.

.

coordenada1 coordenada2 coordenadaN

Nota el separador entre coordenadas puede ser una coma, un tabulador o un espacio en blanco.

Ejemplo.

2

6

1 2

2 1

3 4

5 6

7 8

0 0

En este caso la dimensión de los puntos es 2, el número de puntos es 6 y de ahí siguen los puntos.

El archivo de salida se guarda las coordenadas de los centroides, así como cada punto con su respectivo centroide.

Presentamos un ejemplo de corrida con 20 puntos entre 0 y 1, numerados de 0 a 19 ver tabla 3.2, con los cuales se calcularon 2 centroides.

Número	X	Y
0	0.5	0.1
1	0.7	0.4
2	0.7	0.9
3	0.8	0.3
4	0	0.7
5	0.3	0.6
6	0.8	1
7	0.3	0.2
8	0.9	0.7
9	0.7	0.6
10	0.4	0.7
11	0.1	0.4

12	0.5	0.4
13	0.9	0.1
14	0.1	0.2
15	0.3	1
16	0.2	0.8
17	0.7	0.1
18	0.5	0.9
19	0.3	0.4

Tabla 3.2 Puntos con que se probo el programa de prueba K-*Means*.

Al ejecutar el programa dio como resultado los centroides (0.72, 0.46) y (0, 0.7) y los conjuntos S:

S1 = {0, 1, 2, 3, 6, 8, 9, 12, 13, 17}

S2 = {4, 5, 7, 10, 11, 14, 15, 16, 18, 19}

En la figura 3.8 se muestra la grafica en donde se presentan los centroides y los puntos.

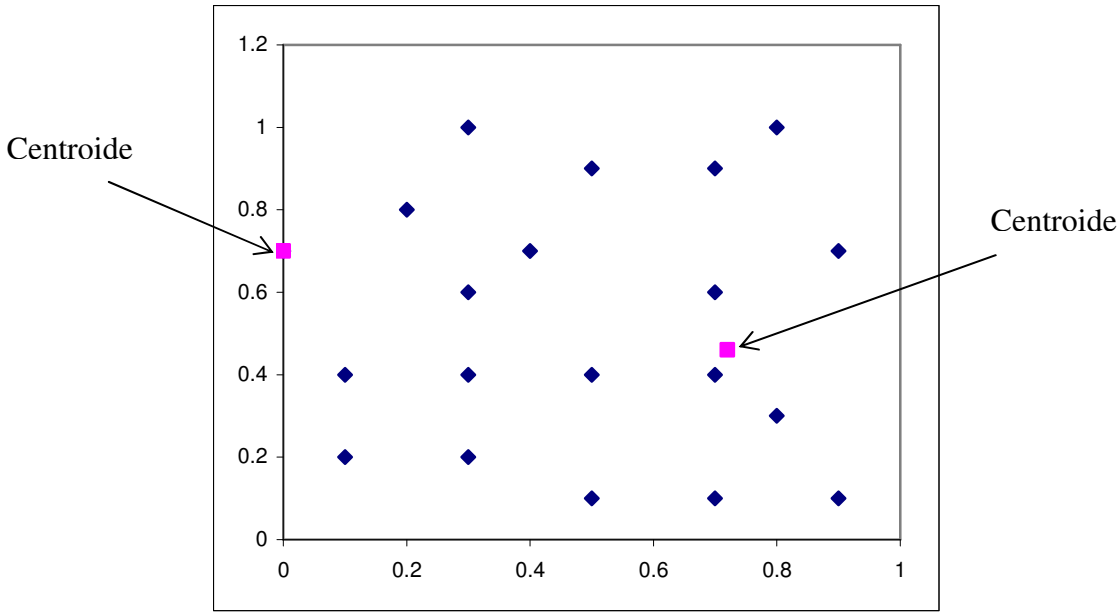


Figura 3.8 Grafica en 2 Dimensiones con 2 centroides y 20 puntos.

El siguiente programa en implementar fue el de la Red de *Kohonen* (ver figura 3.8), el funcionamiento es similar al anterior, además del archivo de patrones ahora también se tiene que abrir o crear uno para la Red de *Kohonen* (ver figura 3.9).

Figura 3.9 Programa de la Red de *Kohonen*.

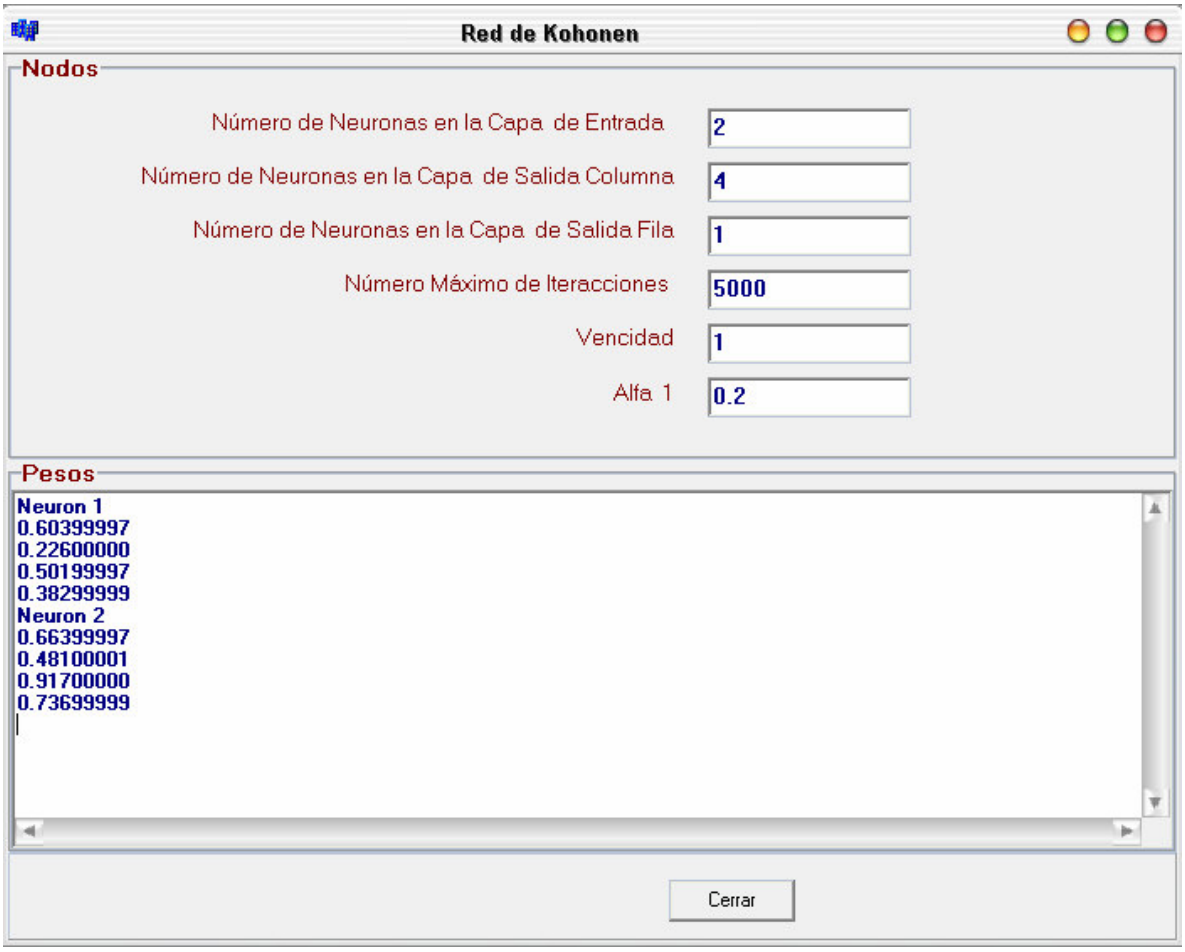


Figura 3.10 Parámetros de la Red y sus Pesos.

Primero se crea o se abre una Red de *Kohonen*, luego se lee el archivo de patrones, para posteriormente ejecutar alguno de los siguientes procesos:

a) Entrenar la red. Se entrena la red tomando como el número de interacciones el valor que se dió al momento de crear la red de *Kohonen*. Una vez terminado el proceso de entrenamiento el programa le pregunta al usuario si desea guardar los cambios ocurridos en los pesos. Estos pesos son guardados en el mismo archivo de la Red.

El entrenamiento se realiza según el algoritmo mostrado en la figura 3.11.

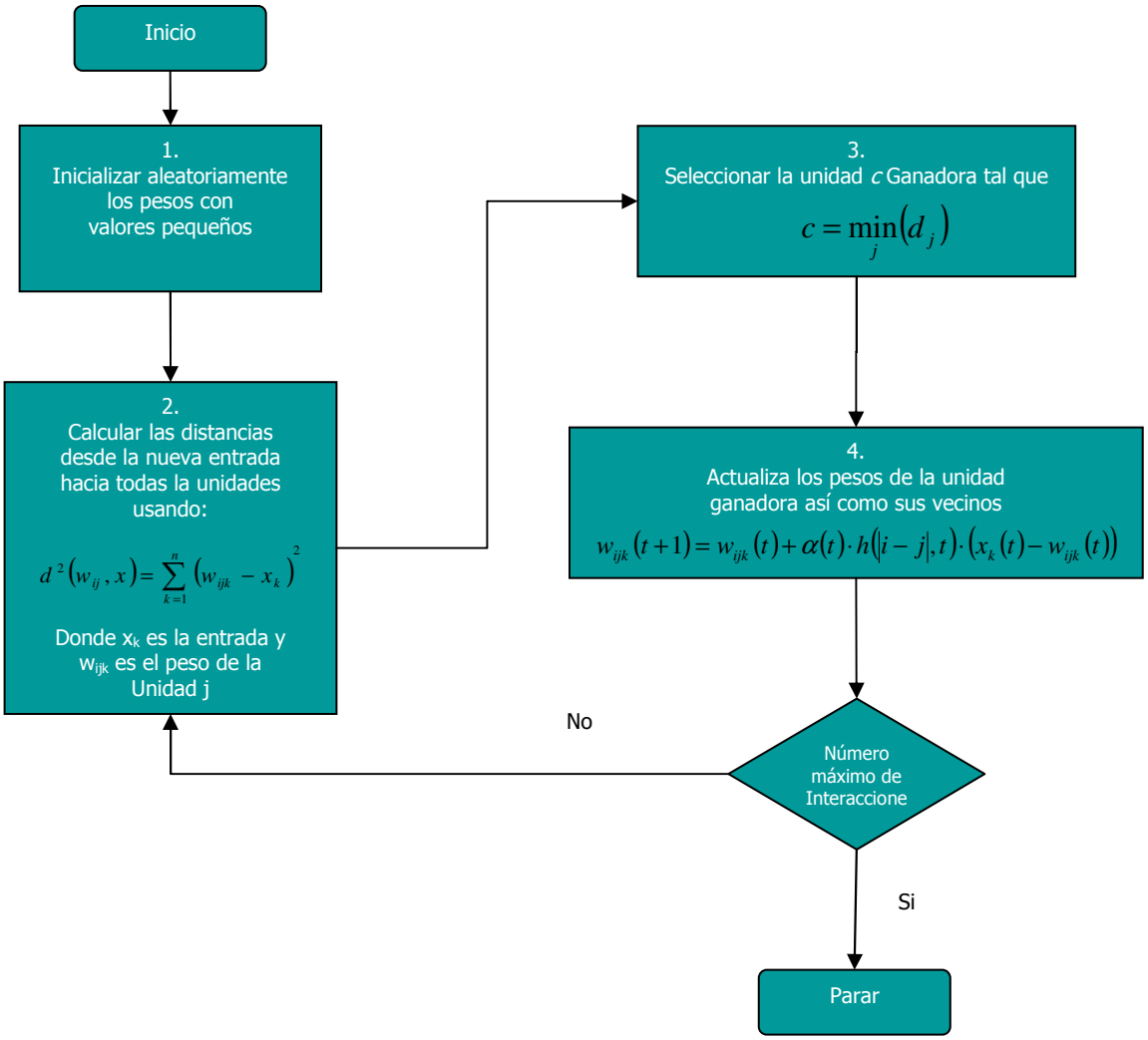


Figura 3.11 Diagrama de flujo del entrenamiento de la red de *Kohonen* [Schalkoff 97].

b) Ejecutar la red. Una vez entrenada la red (aunque puede realizarse el proceso sin entrenar), se puede probar la red, para esto se tiene que abrir un archivo de patrones (que puede ser el mismo que se utilizó para el entrenamiento u otro archivo), o se introducen los valores que se usaran. En la ventana aparecen varios botones: "Agregar Renglón", si deseamos agregar otro vector para la entrada. "Eliminar Ultimo Renglón", si se desea eliminar el último vector para la entrada de la red. Cabe señalar que no lo elimina del archivo solo lo elimina del gris de la pantalla. Y al final aparece "Calcular" que aquí realiza la ejecución de la red para posteriormente presentarnos por cada vector de entrado quien fue la neurona ganadora (ver figura 3.12).

Figura 3.12 Ventana del proceso ejecutar del programa de Red de *Kohonen*.

El archivo de patrones lleva el mismo formato usado en el programa de K-Means descrito anteriormente, para el de la Red de *Kohonen* es el siguiente formato.

Número de Neuronas en la Capa de Entrada

Número de Neuronas en Columnas en la Capa de Salida

Número de Neuronas en Filas en la Capa de Salida

Número de Interacciones

Valor de la Vencidad

Valor de Alfa1

Valores de los Pesos.

.

.

Valores de los Pesos

Ejemplo

2

2

2

5000

1

0.20

0.67900002 0.42604348

0.40099999 0.42604348

0.40000001 0.53042459

0.82800001 0.53042459

Mencionamos que también se hicieron unas prueba con un componente para c++Builder SDL Delphi Component Suite New Release 7.2, de la empresa Software Development Lohninger, que incluye la Red de Kohonen, para comparar resultados con nuestro programa desarrollado. Este componente es operable al 100% dentro del entorno de desarrollo del c++Builder, pero fuera del entorno varias funciones son deshabilitadas, mandando varias

ventanas de anuncio indicando si deseas adquirir la licencia. Más referencia visitar la página de la empresa [SDL 03]

3.2 Software Reconocimiento de Caracteres.

Esta es la aplicación principal. Cuando se abre una imagen automáticamente se calcula la matriz binaria (0 – blanco, 1- Negro) de la imagen (ver figura 3.11), para posteriormente aplicar los siguientes procesos.

Nota: Por ahora se tiene que dar los puntos de segmentación manualmente, falta que se una con el proyecto de [Navarrete 02], ya que el proyecto HAWOSS fue realizado en Java, y este en c++Builder.

a) Primer Paso: Segmentación. Una vez dados los puntos de segmentación, se extraen las porciones de la matriz de punto y se almacenan en una lista de matrices binarias ver figura 3.13.
b) Segundo Paso: Recorte. Con la lista de matrices ahora se eliminan los espacios en blanco tanto arriba, abajo, izquierda como a la derecha, como se muestra en la figura 3.14.
c) Tercer Paso: Normalización. Una vez recortada la matriz tenemos que normalizarla, para esto se usa el algoritmo de Güdsen [Aguilar & Toledo 95]. Este consiste en escalar cada columna de la imagen (y posteriormente cada renglón) a un tamaño fijo. Se define WA como el vector de elementos de la matriz binaria de entrada y el vector WB como el vector de salida. A su vez se define un vector con dimensión WA*WB, para almacenar WB veces cada elemento de WA. Después se forman sub-grupos de WA tamaños y se calcula la suma de su contenido. Si esta suma

supera un umbral establecido se considera como 1, en caso contrario como 0 (ver figura 3.15).

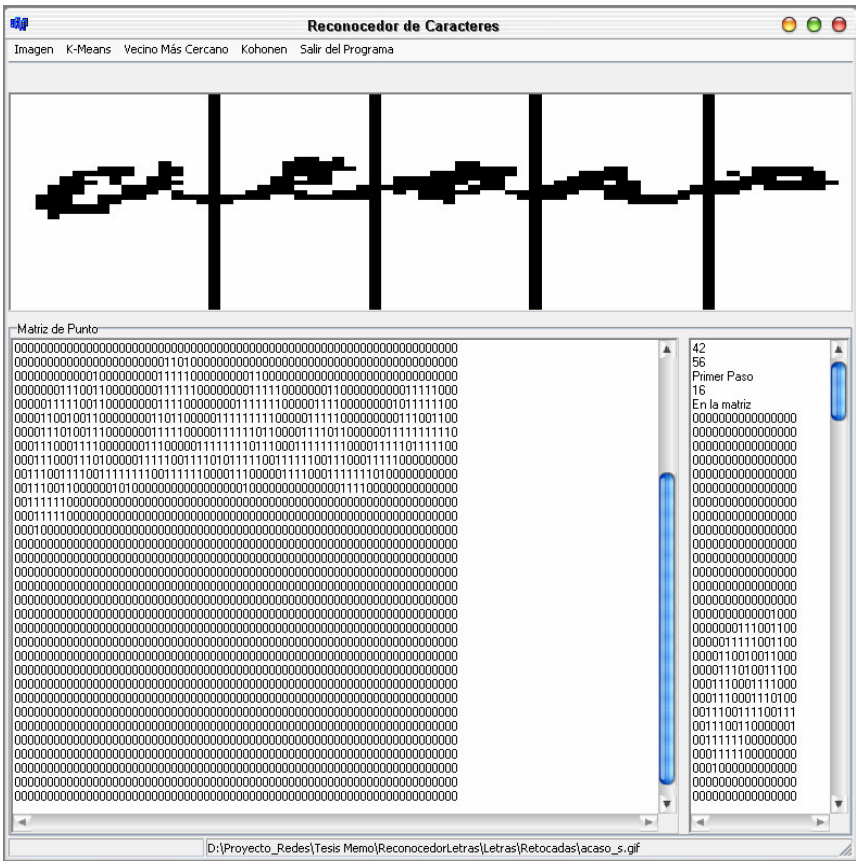


Figura 3.13 Ejecución del programa Reconocedor de Caracteres.

00000000001000
00000111001100
00011111001100
00110010011000
00111010011100
01110001111000
01110001110100
11100111100111
11100110000001
11111100000000
01111100000000
01000000000000

Figura 3.14 Matriz de puntos recortada.

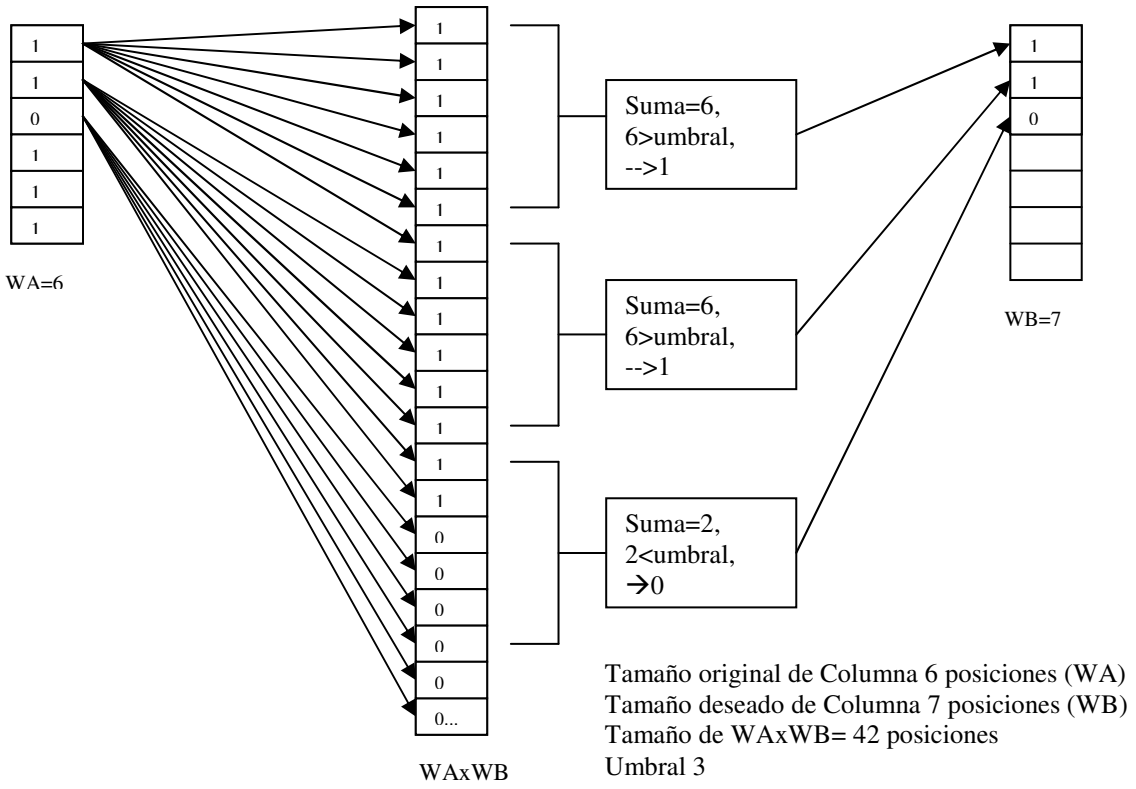


Figura 3.15 Diagrama del funcionamiento del Algoritmo de *Güdsen* [Aguilar & Toledo 95].

Se probaron varias configuraciones (Renglón X Columna), para determinar la más optima, la matriz resultante fue de 12 renglones X 18 columnas, la figura 3.14 b), que es la matriz resultante después de la normalización.

000000000001000	0000000000000010000
000001110011100	0000001111000011100
000111110011100	0000111111000011100
001100100110000	0011100010011100000
001110100111000	0011111010011111000
011100011110000	0111100001111100000
011100011101000	0111100001111011000
111001111001111	1111001111110011111
111001100000001	1111001110000000001
111111000000000	1111111100000000000
011111000000000	0111111100000000000
010000000000000	0110000000000000000
a)	b)

Figura 3.16 Resultado del Segundo al Tercer Paso en la Matriz de Puntos de la Letra “a”.

Una vez realizada las tres operaciones se puede crear un archivo de patrones o agregarlo a uno existente (en este caso debe de coincidir el número de dimensiones).

Una vez creado los archivos de patrones (los cuales servirán para calcular los centroides por medio de K-Means, Entrenar la Red de Kohonen, y de prueba para vecino más cercano y la propia Red de Kohonen), se hacen las pruebas requeridas que se muestran en el siguiente capitulo.

3.2 Software de apoyo para la Creación de Archivos.

Una vez realizado la segmentación, cortado y normalización de los caracteres que componen cada palabra en las imágenes, éstas son almacenadas en un archivo. Para la creación de los archivos de patrones se tiene que ir tomando cada caracter y agregarlo al archivo de patrones que se requiere. Para esto realizamos el programa que se muestra en figura 3.17.

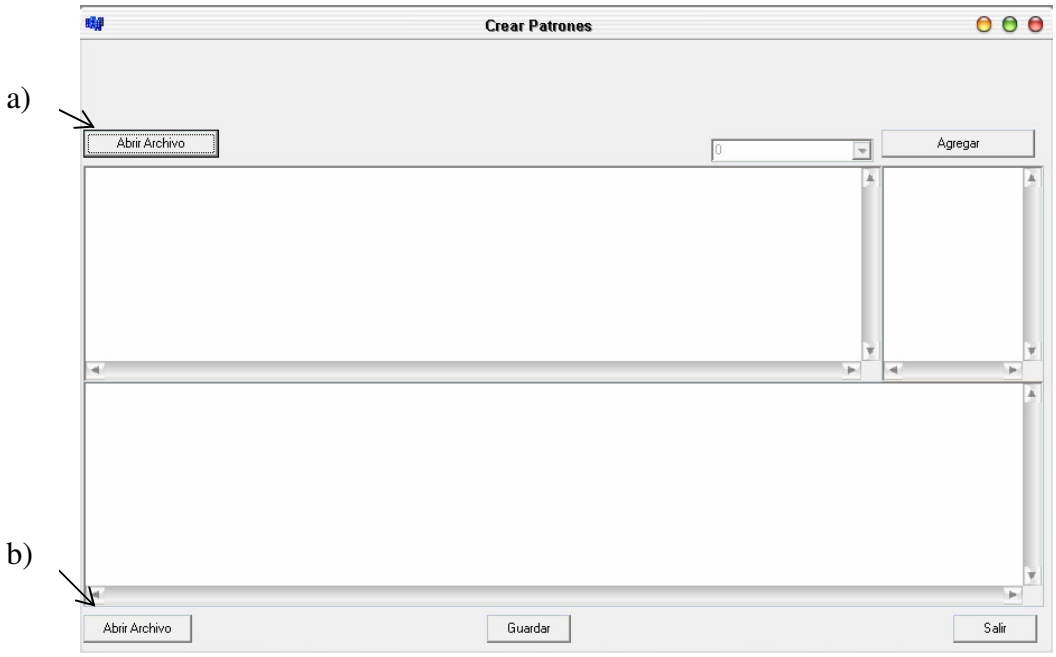


Figura 3.17 Programa para la Creación de Archivos de Patrones.

El programa funciona de la siguiente manera, si se da *clik* en el botón superior izquierdo (a) "Abrir Archivo" se abre un archivo de patrones, y el programa da por hecho que se creará un nuevo archivo de patrones, por lo cual deshabilita el botón inferior izquierdo (b). Este botón (b) sirve para abrir un archivo de patrones al cual se le requiere agregar más patrones, por lo tanto si lo que se quiere es agregar más patrones a un archivo, primero hay que darle clic a este botón (b) y después abrir darle clic al botón (a) que esta en la parte superior izquierda.

Una vez abierto un archivo de patrones el programa muestra lo siguiente (ver figura 3.18):

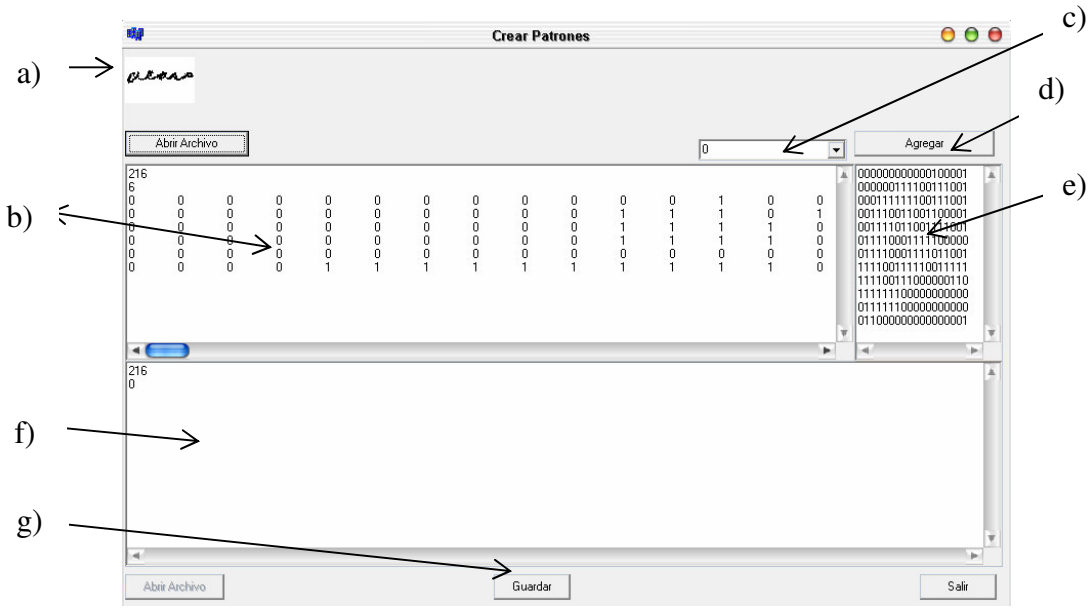


Figura 3.18 La visualización del programa una vez abierto el archivo de patrones.

En este caso el programa mostrará, si existe, la imagen (archivo Gif) de la palabra (a), todos los caracteres que componen a la palabra (b), una lista desplegable (c) que contiene la lista de los caracteres, en la cual se puede seleccionar un caracter para ser visualizado en el campo memo (e). Con el botón que se encuentra en la parte superior derecha (d) "Agregar" se agrega el caracter que se encuentra seleccionado en la lista desplegable, y lo agrega al

campo memo (f). Si desea puede ir abriendo más archivos con el botón (a) e ir agregando más caracteres al campo memo (f). Cuando el usuario lo requiere puede ir salvando el contenido del campo memo (f) a un archivo con el botón (g) “Guardar”, el programa se verifica que si el archivo donde se desea grabar existe, entonces se le solicita al usuario la confirmación de sobre escritura como se muestra (ver figura 3.19).

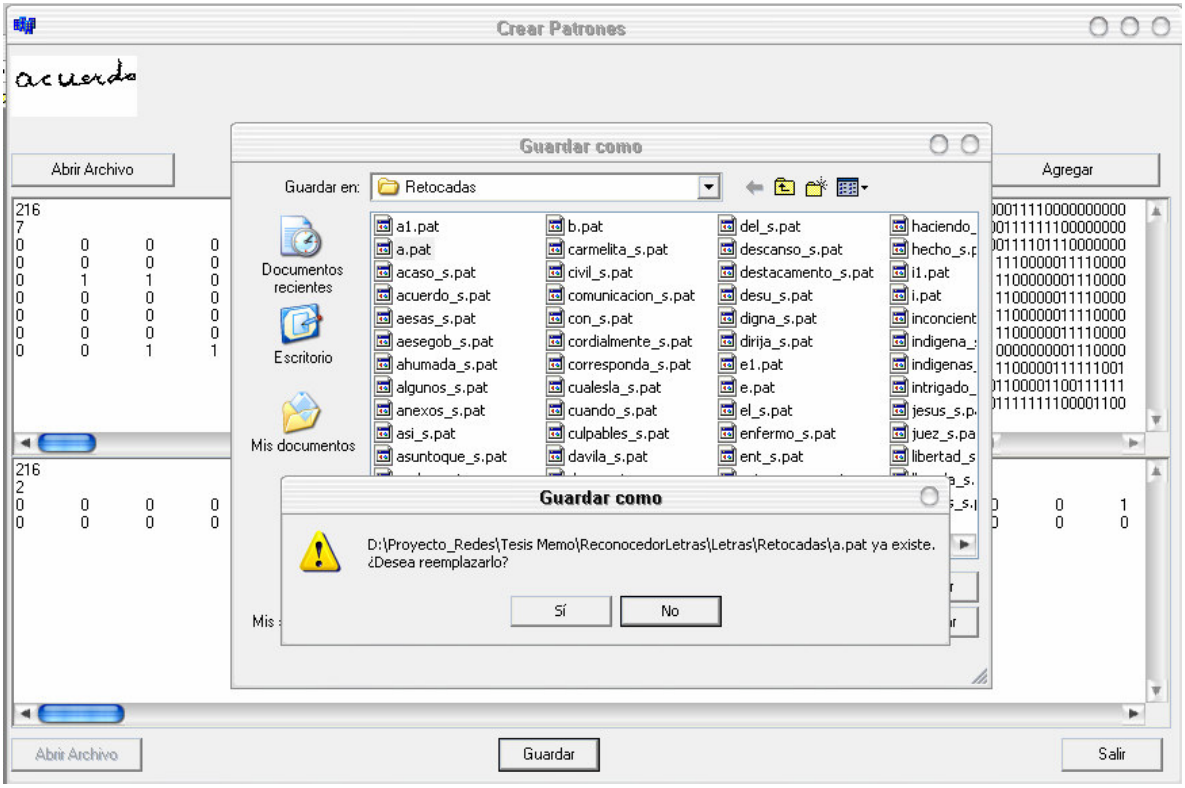


Figura 3.19 Solicitud al usuario si desea sobre escribir el archivo.