UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERIA ELECTRÓNICA

RECONOCEDOR ÓPTICO DE CARACTERES MANUSCRITOS (ROCM) POR MEDIO DE REDES NEURONALES

TRABAJO DE GRADUACIÓN PARA OPTAR AL GRADO DE INGENIERO EN ELECTRÓNICA

PRESENTADO POR:
FLORES GÓMEZ, MIGUEL EDUARDO

Ciudadela Don Bosco, Marzo de 2004.

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERIA ELECTRÓNICA

RECONOCEDOR ÓPTICO DE CARACTERES MANUSCRITOS (ROCM) POR MEDIO DE REDES NEURONALES

Ing. Eduardo Rivera
Asesor

Ing. Edgardo Zeledón
Jurado

Ing. Ana Daysi
Jurado

Ing. Jorge López
Jurado

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERIA ELECTRÓNICA

RECONOCEDOR ÓPTICO DE CARACTERES MANUSCRITOS (ROCM) POR MEDIO DE REDES NEURONALES

Ing. Federico Huguet
Rector

Lic. Mario Olmos
Secretario General

Pbro. Victor Bermúdez
Vicerrector Académico

Ing. Carlos Bran
Decano de la facultad de Ingeniería.

Ciudadela Don Bosco, Marzo de 2004.

Agradecimientos:

Hay tanto que agradecer a tantas personas que no puedo enumerarlas a todas si hay alguien que se me olvida mencionar mis disculpas, pero espero hacer tanto por otros así como otros han hecho por mí.

Mis mayores agradecimientos son para el Maestro, mi Guía y Redentor, quien me dio fuerzas y ánimo en los momentos difíciles, serenidad y paciencia para sobrellevar las cargas, fuerza y valentía para enfrentar los problemas. Sin Él nada habría sido posible.

Para mis padres que me iniciaron en el estudio, me ayudaron a ser independiente y a trabajar para alcanzar mis metas.

Para mi novia por ayudarme en mis faenas, por su apoyo en tantos aspectos tanto materiales como espirituales. Para su familia que me ha soportado todo este tiempo sin quejarse, tratándome como a alguien más de la familia.

También quiero mencionar a mis profesores por sus enseñanzas y a mis compañeros de estudio, estos últimos deben saber que no hay meta inalcanzable y que el esfuerzo recibe su premio aunque no siempre de la manera que esperamos.

Gracias a todos, y que al final de esta senda de la vida nos encontremos todos junto al fuego reunidos por el Señor....

INDICE

1. INTRODUCCIÓN

A lo largo del documento se presentan las partes del trabajo de graduación, el Reconocedor Optico de Caracteres Manuscritos Usando Redes Neuronales, las cuales son:

- El segmentador de caracteres.
- La Interfaz Gráfica de Usuario.
- El módulo de la red neuronal a ocupar.
- Preprocesamiento de la imagen (ampliación y vecino 8).
- El identificador, que traduce la salida activada de la red en un caracter de texto.

Se muestra la teoría, funcionamiento, y aspecto de los elementos anteriormente mencionados. Cada módulo, del reconocedor está descrito en detalle y se muestra un flujograma del funcionamiento de cada uno de ellos. Sus alcances y limitaciones también son descritos.

También se muestran módulos que complementan el funcionamiento del OCR como son:

- El módulo de ayuda para manejar el programa.
- El creador de nuevos usuarios
- Reglas de escritura.

Estos módulos también son descritos y explicados en el documento. Aparecen en la sección donde se describe la Interfaz Gráfica de Usuario.

En los anexos también se muestra una tabla donde se resumen los resultados de los reconocimientos hechos con el OCR durante la etapa de prueba y afinamiento de la red, la ayuda del programa y los códigos de los módulos ocupados para el reconocedor óptico de caracteres manuscritos.

2. OBJETIVOS, ALCANCES, LIMITACIONES Y VALIDACIÓN DE RESULTADOS

2.1 Objetivos

2.1.1 Objetivo General:

Elaborar un programa que convierta la información manuscrita, contenida en un archivo de imagen, a formato texto para que el usuario pueda manipularla con algún procesador o editor de texto convencional; dicha conversión será hecha con un alto grado de fiabilidad usando redes neuronales para la identificación de caracteres.

2.1.2 Objetivos específicos:

- Proponer el lenguaje Perl como una alternativa en la creación de programas útiles.
- Crear una aplicación práctica y funcional de inteligencia artificial.
- Usar redes neuronales para la identificación de caracteres.
- Crear una interfaz gráfica en el lenguaje Perl para el manejo y control del proceso.
- Lograr que el programa sea capaz de "aprender" la forma de escribir del usuario para mejorar su eficacia.
- Crear un archivo en formato texto con la información que antes se encontraba manuscrita en el archivo de imagen.

2.2 Alcances y limitaciones

2.2.1 Alcances:

- Lograr la conversión de información manuscrita a formato texto con un alto grado de fiabilidad.
- Brindar una alternativa a la introducción de datos de forma manual a la computadora.
- Usar un lenguaje de programación de libre distribución para la creación del programa.
- Crear una aplicación práctica en Perl que muestre su potencial como lenguaje de programación alternativo a otros más populares.
- Presentar una interfaz gráfica agradable y fácil de manejar para el usuario.
- Crear una aplicación práctica de inteligencia artificial.
- La red neuronal será capaz de "aprender" la forma de escritura del usuario aumentando con ello las capacidades de identificación del programa.
- Después de la identificación realizada por el programa, el usuario podrá realizar las correcciones necesarias en el texto a través de la ventana que muestra los resultados.
- Realizará muestreos alrededor del área donde se supone existe una letra para identificar aquellos caracteres cuyas partes estén separadas por distancia pequeñas.
- Identificará las letras del alfabeto español, los dígitos, signos de puntuación, admiración, interrogación, vocales tildadas.

- El programa se encargará de almacenar los pesos personalizados de un usuario en un archivo para identificar su forma de escritura, dichos pesos podrán ser o no actualizados cada vez que se use el programa según decida el usuario.
- Se pueden crear múltiples perfiles para los distintos tipos de letra que tenga el usuario.

2.2.2 Limitaciones:

- El tamaño máximo y mínimo de las letras estará restringido
- No identificará fórmulas matemáticas, tablas o ecuaciones.
- La línea de texto manuscrito debe ser lo más horizontal que se pueda para no afectar la eficacia del reconocimiento.
- Cada letra deberá estar separada de cualquier otra letra por algún espacio en blanco, tanto en dirección horizontal como vertical, por lo cual no deben estar dos letras unidas o no se asegura una correcta identificación del caracter, ya que el programa a desarrollar será enfocado a caracteres espaciados, aunque no se descarta que pueda identificar caracteres unidos (letra de carta), los cuales podrá identificar mejor si se encuentran bastante espaciados entre sí, aunque siempre estén unidos por el mismo trazo.
- Trabajará en el sistema operativo Windows 95 o superior.
- Perl[1] deberá estar instalado para que el programa de OCR funcione.
- El programa no se encargará de digitalizar la imagen con la que se va a trabajar.
- El formato del archivo de salida será de solo texto (*.txt).
- El formato del archivo imagen de entrada estará limitado por los soportados por el programa Perl, (hasta ahora se ha comprado su funcionabilidad con archivos .bmp, .gif).
- Los requisitos del sistema serán, por lo menos, los mismos que los del lenguaje Perl[2].
- No se asegura un alto grado de acierto en la identificación de caracteres a máquina o de imprenta, a menos que la red sea entrenada adecuadamente para ello.

2.3 Validación de resultados

Se realizarán pruebas de aprendizaje e identificación de varios usuarios y se tabularán los resultados para cuantificar su efectividad.

Con los datos obtenidos se podrá apreciar el porcentaje de errores y aciertos del sistema, con lo que se podrán hacer comparaciones con los porcentajes anunciados por otros identificadores de caracteres del mercado.

Además se realizará una prueba de aprendizaje e identificación en el momento de la evaluación.

---

[1] Para una mejor descripción del programa, requisitos del sistema, licencia, y limitaciones consultar la referencia 6.

[2] Windows 9x o superior, 55 MB libres de disco duro, MSI installer, Internet Exploerer 5 o superior. Sacado de la referencia 6.

3. PROGRAMA DE RECONOCIMIENTO DE CARACTERES MANUSCRITOS

El programa se compone de diversas etapas como se muestra en la figura 3.1:

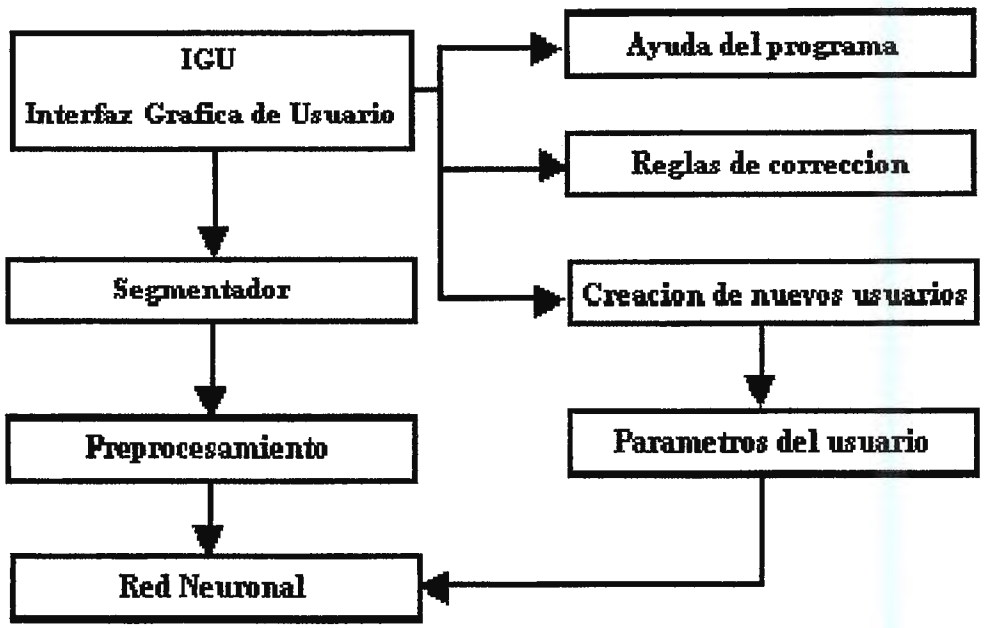


Figura 3.1 Etapas del Programa Reconocedor Optico de Caracteres Manuscritos.

El funcionamiento general del programa es:

- La Interfaz Gráfica de Usuario se encarga de dirigir todo el proceso, llamando a cada rutina o función según corresponda, es ella quien llama al segmentador, la ayuda del programa, al módulo de la red neuronal y los demás módulos restantes, según lo vaya solicitando el usuario. Además es ahí donde se pueden modificar los parámetros del programa.
- El segmentador es quien divide la imagen, proporcionada por el usuario, en caracteres separados para poder ser identificados, además cuenta la cantidad de caracteres encontrados así como el número de líneas de la imagen.
- La red neuronal, es el identificador de los caracteres, es ahí donde llegan los datos obtenidos por el segmentador y son clasificados de acuerdo a sus características.
- La personalización del programa para nuevos usuarios se utiliza para "adiestrar" al programa, sirve no solo para personalizar los pesos de la red neuronal, sino que además proporciona parámetros útiles al segmentador, para que éste pueda realizar su trabajo con mayor precisión.
- En los archivos de parámetros de los usuarios es donde se guardan los elementos configurables del programa, como son: los pesos de la red neuronal para cada usuario, el caracter relacionado con cada salida de la red y los parámetros del segmentador.

- La ayuda del programa proporciona información acerca de los elementos del ROCM, su funcionamiento, así como algunos problemas en el reconocimiento y sus posibles soluciones.

3.1 IGU (Interfaz Gráfica de Usuario)

3.1.1 Descripción de la Interfaz Gráfica de Usuario.

En la figura 3.1.1 se muestra la IGU del programa. Como en cualquier aplicación de Windows, la IGU (o GUI en inglés), es una de las partes más importantes del programa, ya que se encarga de servir de intermediario entre el usuario y el programa.

Para el caso del ROCM la interfaz es proporcionada aprovechando el módulo TK[3] de Perl, el cual permite crear los elementos más comunes como botones de comando, ventanas, cajas de diálogo, etiquetas, y otros controles más, además es este módulo el que se encarga de llamar a la rutina o función adecuada para responder a los eventos generados por el usuario. Por ejemplo llamar a la función para guardar un archivo, invoca la función adecuada para la creación de un nuevo usuario, la ayuda del programa, la corrección automática de los caracteres reconocidos.

El entorno gráfico que es creado usando TK responde, como cualquier otra aplicación nativa de Windows, a los eventos del usuario por teclado y ratón. Se han adicionado algunos elementos más a la interfaz que no necesariamente aparecen en todas las aplicaciones pero que son útiles, tales como el aparecimiento de balones o globos de ayuda cuando el puntero del ratón se posiciona por más de 0.35 seg. encima de un icono de la barra de herramientas, dicho balón menciona la función del icono para servir de guía.

Además para la corrección de los caracteres de reconocidos se tiene una herramienta para buscar y reemplazar texto.

La IGU posee elementos que les serán familiares a las personas que tengan cierto tiempo manejando aplicaciones de Windows, por ejemplo los cuadros de diálogo para abrir y guardar archivos son los mismos, por lo que no les será difícil interactuar con el programa.

Se compone de varias partes:

- Menú del programa.
- Barra de herramientas
- Ventana de texto principal
- Ventana de texto secundaria.

[3] Ver referencia 7.

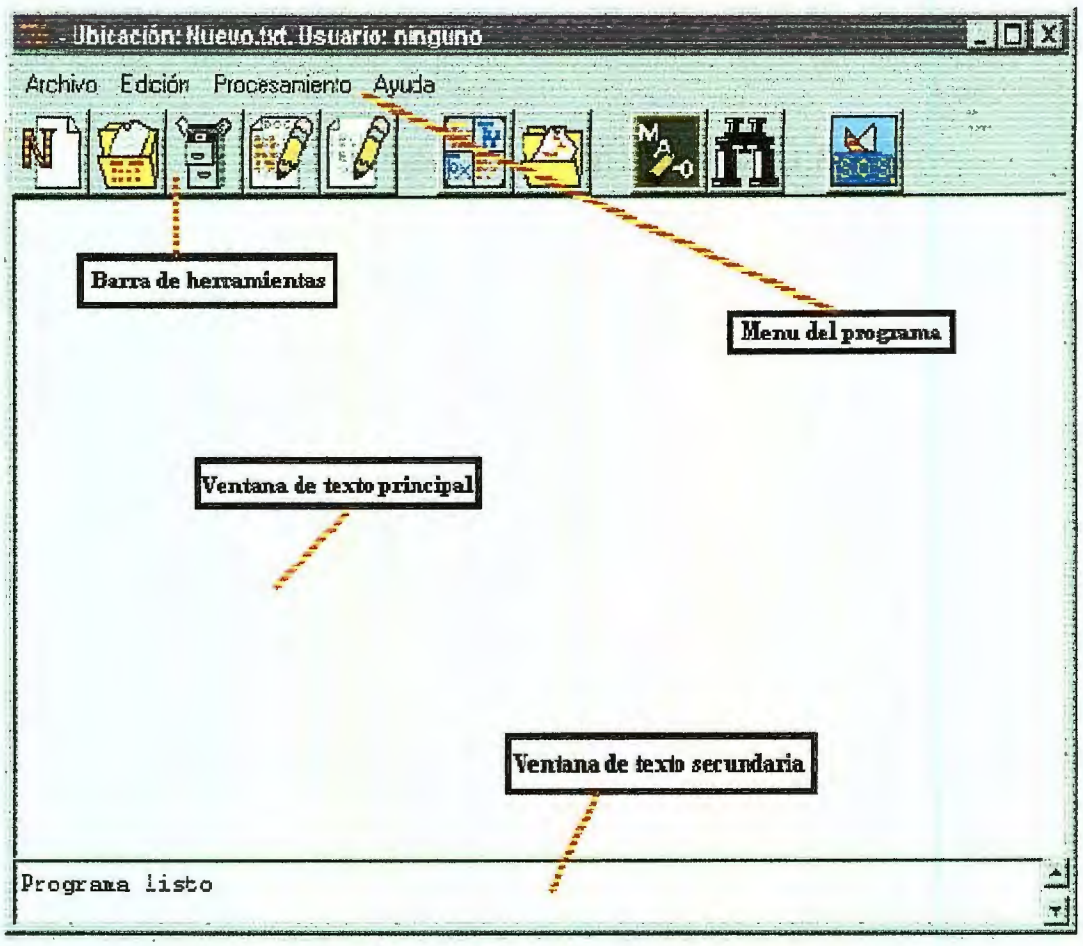


Figura 3.1.1.1. Interfaz Gráfica de Usuario, pantalla de trabajo.

El menú del programa es desde donde se controlan todas las funciones del reconocedor, tales como: abrir archivos de texto, crear archivos de texto, variar los parámetros del segmentador, abrir un archivo de imagen, llamar la configuración de un usuario ya existente, consultar la ayuda del programa, etc.

Su funcionamiento es muy similar al de otras aplicaciones para el sistema operativo Windows, incluyendo el uso de atajos para llamar funciones muy usadas (por ejemplo con Alt + a se llama al cuadro de diálogo para abrir un archivo de texto).

El cuadro de herramientas son maneras fáciles de acceder a las funciones más ocupadas con solo mover el ratón y pulsar una vez sobre el icono. Cada botón del cuadro de herramientas posee su propia descripción breve que se muestra al mantener el puntero del ratón inmóvil sobre él por aproximadamente 0.35 segundos (el tiempo se puede variar ajustando el programa). Actualmente se muestran las siguientes funciones en el cuadro de herramientas (aparecen de izquierda a derecha en la figura 3.1.1.1.): crear un nuevo archivo de texto, abrir un archivo de texto, guardar un archivo de texto, crear un nuevo usuario,

llamar la configuración de un usuario ya existente, cambiar los parámetros del segmentador y abrir un archivo de imagen para poder procesarlo.

El cuadro de texto principal es donde aparecerán los caracteres identificados por el reconocedor, desde ahí el usuario podrá manipular el resultado a su gusto y posteriormente guardarlo en un archivo de texto (.txt), el cual podrá ser leído por cualquier otra aplicación que maneje texto.

El cuadro de texto secundario se utiliza para mostrar los mensajes de estado o advertencias de parte del programa, así como para consultar que acciones se han realizado, como cambio de parámetros del segmentador, abrir un archivo, el nombre del archivo de imagen que se ha trabajado.

Para acceder a los parámetros de un usuario ya creado se hace pinchando sobre el icono en la barra de herramientas llamado *usuario*, con la combinación de botones *Alt* + *u* o desde *archivo→usuario*, de cualquiera de las tres maneras se presenta un cuadro de diálogo como el mostrado en la figura 3.1.1.2 donde se ingresa el nombre del usuario ya creado. Si el usuario existe se presenta un mensaje de bienvenida en el cuadro de texto secundario, de lo contrario manda un mensaje indicando que el usuario no existe.

Figura 3.1.1.2. Cuadro de diálogo para personalizar al programa con los parámetros de un usuario ya creado.

3.1.2 Programación de la IGU

El módulo TK posee ciertas facilidades para la creación de IGU, tales como:

- Que dicho módulo se encarga de manejar los eventos y llamar a la función solicitada por el evento.
- No es necesario especificar todos los parámetros que posee cada *widget*[4] (o control), para que funcione adecuadamente.
- Se pueden modificar partes de un texto en una caja de texto para que presente distinto formato haciendo uso de etiquetas (*tag*[5]), con lo que cada parte puede tener un formato diferente de otra. Además se puede asociar acciones a estas etiquetas para ejecutar

[4] *widget* es el nombre que se le da en Perl a los controles que se presentan en la IGU, tales como: botones de comando, etiquetas y ventanas.
[5] *tag* con este nombre se distingue una porción de texto seleccionada a la que se le asigna un nombre y formato propios.

alguna acción cuando se produce algún evento en ellas, por ejemplo presionar un botón del ratón, pudiendo servir como vínculos.

El diagrama de flujo para la IGU es el siguiente:

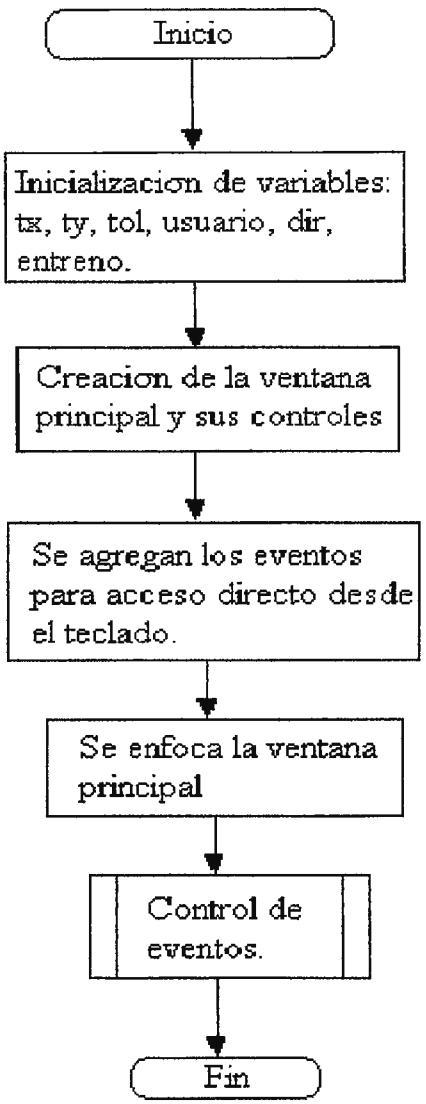


Figura 3.1.2.1 Diagrama de flujo de la IGU.

A continuación se describen los elementos del diagrama de flujo:

- Inicialización de variables:

Nombre de la variable:	Valor con que se inicializa:	Se ocupa para:
$tx	40	Establecer el ancho de la ventana de muestreo del segmentador.
$ty	50	Establecer la altura de la ventana de muestreo del segmentador.

$tolx	3	Indicarle al segmentador el valor de tolerancia (separación entre dos pixeles en negro) máximo en la dirección x.
$toly	3	Indicarle al segmentador el valor de tolerancia (separación entre dos pixeles en negro) máximo en la dirección y.
$usuario	"hinguno"	Definir que no se han escogido los parámetros de algún usuario ya creado.
$dir	directorio actual de trabajo	A partir del directorio actual de trabajo el programa busca las carpetas donde se encuentran archivos que son ocupados durante su funcionamiento. Por ejemplo archivos de ayuda, usuarios, pesos, etc.
$entreno	"no"	Saber si el segmentador funcionará en modo de entrenamiento o trabajo normal.

Tabla 3.1.2.1. Variables inicializadas durante el primer paso de la IGU.

- Creación de la ventana principal y sus controles (ver figura 3.1.1.1.): La ventana principal contendrá varios controles como los botones de comando que se usan como barra de herramientas, el menú del programa, etiquetas, y las dos cajas de texto. La caja de texto principal es donde se presentan los resultados de la segmentación y la caja de texto secundaria es donde aparecen los mensaje relacionados a lo hecho en el programa como: indicación de la cantidad de letras y líneas encontradas durante la segmentación, mensajes de bienvenida cuando se introduce un nombre de usuario válido y los valores de *$tx*, *$ty*, *$tolx* y *$toly* que se establecen al variar los parámetros del segmentador.
- Se agregan los eventos para acceso directo desde el teclado[6]: Hay combinaciones de teclas ya definidas por TK como por ejemplo: Ctrl + c para copiar, Ctrl + v para pegar. Pero es necesario definir las que se refieren a las rutinas propias del programa como Alt + a para abrir un archivo, Alt + b para borrar todo lo que se encuentra en la ventana principal, Alt + n para crear un nuevo archivo de texto.
- Se enfoca la ventana principal: Con el objetivo que la ventana principal sea la aplicación activa.
- Control de eventos: A partir de ese momento el programa entra en un lazo durante el cual captura los eventos que se realizan en la ventana principal para poder llamar a las rutinas utilizadas. Las rutinas son descritas con detalle en la sección siguiente.

La IGU ocupa otros módulos adicionales para complementar sus funciones, uno para manejar la ayuda del programa, otro para el asistente que crea nuevos usuarios y uno más para realizar la corrección automática. Dichos módulos serán explicados en las secciones 3.1.3., 3.1.5. y 3.1.7. respectivamente.

[6] En el anexo A.1.2 se muestra un listado de las teclas de acceso directo disponibles para el programa.

3.1.2.1 Rutinas ocupadas en la IGU.

Las rutinas ocupadas en la IGU para el menú *Archivo* son las siguientes:

- **AbrirArchivo:** presenta el cuadro de diálogo para seleccionar un archivo de texto que se desea mostrar en la caja de texto principal. El cuadro de diálogo regresa la dirección o camino del archivo a abrir, la información del archivo es mostrada, en la barra de título y en la caja de texto secundaria aparece el nombre del archivo. Dicha dirección se guarda en la variable *$direcciondelarchivo.*
- **GuardarArchivo:** Si el nombre del archivo a guardar es 'Nuevo.txt" indica que es la primera vez que se guarda el archivo de texto por tanto muestra el cuadro de diálogo para guardar un archivo, si el valor regresado por el cuadro es distinto a un valor nulo entonces se guarda el contenido de la caja de texto principal en un archivo de texto con el nombre especificado, además el nombre del archivo es desplegado en la barra de título y en la caja de texto secundaria y guardada en la variable *$direcciondelarchivo.* Si el nombre es distinto de 'Nuevo.txt" entonces guarda la información sin mostrar el cuadro de diálogo debido a que ya posee un nombre el destino (*$direcciondelarchivo*)y guarda la información, además de presentar el mensaje de guardar en la caja de texto secundaria.
- **GuardarArchivoComo:** Se asigna a *$direcciondelarchivo* el nombre 'Nuevo.txt" y se llama a la rutina **GuardarArchivo**.
- **NuevoArchivo:** Borra el contenido de la ventana de texto principal, establece *$direcciondelarchivo* a 'Nuevo.txt", y se configura el título de la ventana para que aparezca el nuevo nombre del archivo.
- **nusuario:** Llama a la función *inicia* del módulo *nusuario* para iniciar la etapa de entrenamiento de la programa para un nuevo usuario. Establece la variable *$entreno* a "si", esto sirve para indicarle al segmentador que función debe ser activada al presionar el botón de *procesar.*
- **usuario:** Se encarga de llamar el archivo que contiene los parámetros del usuario solicitados. Muestra un cuadro de diálogo donde se introduce el nombre del usuario, el nombre del archivo se forma de esta manera: "*nombre del usuario.par*", dicho archivo deberá encontrarse en la carpeta usuarios del programa. Si el usuario existe, en el título de la ventana y en el cuadro de texto secundario aparece el nombre del usuario, sino existe aparece un mensaje en el cuadro de texto secundario para alertar al usuario, aparte de producir un sonido de advertencia.
- **gusuario:** Guarda los parámetros del usuario (tamaños de ventana del segmentador y tolerancias), con el objeto de darle al usuario un mayor control y flexibilidad en el proceso de reconocimiento. Reescribe el archivo donde se encuentran los parámetros del usuario con los que se encuentren en este momento en el programa.
- **salir:** Destruye la ventana principal terminando con ello la ejecución del programa.

Las rutinas ocupadas para el menú *Edición* son las siguientes:

- **SelTodo:** Selecciona todo el texto que aparece en la caja de texto principal.
- **BorrarTodo:** Borra el contenido de la caja de texto principal.
- **Reemplazar:** Llama al cuadro de diálogo para reemplazar el texto de la ventana principal. En el reemplazo se puede escoger entre ir reemplazando individualmente las coincidencias, pidiendo en cada ocasión la aprobación del usuario para realizar el reemplazo, o que reemplace todas las coincidencias a la vez, ver figura 3.1.2.1.1.

Las rutinas para el menú *Procesamiento* se describen a continuación:

- **ParSeg:** Muestra un cuadro de diálogo para establecer los parámetros del segmentador. Al iniciar el programa estos parámetros tienen ciertos valores predefinidos (ver tabla 3.1.2.1). Al presionar el botón aceptar llama a la rutina *rutParSeg,* que se encarga de obtener los parámetros ingresados.
- **rutParSeg:** Con esta rutina obtengo los parámetros que se ingresan en el cuadro de diálogo mostrado por *ParSeg*, los valores de los parámetros obtenidos se muestran en la caja de texto secundaria.
- **AbrirIm:** Llama el cuadro de diálogo para abrir un archivo de imagen (.bmp o .gif), con esta rutina se obtiene el nombre del archivo y su camino, los cuales se usarán como argumentos para llamar a la rutina *MostrarI* que se encuentra en el módulo de segmentación. Debido a que Perl trabaja en sistemas Linux es necesario "traducir" la dirección que se obtiene a una que sea válida para el sistema Windows, es decir, se debe cambiar el "/" que regresa el cuadro de diálogo de abrir archivo por "\". Por ejemplo C:/Windows/imagen/dibujo.bmp pasa a ser C:\Windows\imagen\dibujo.bmp.
- **Corrige:** Llama al módulo *reglas.pm*, para aplicar ciertas reglas de escritura común en el texto que se encuentra en la ventana principal, por ejemplo que aparezcan mayúsculas mezcladas con minúsculas, y cambiarlo por los caracteres que parecieran tener una mayor concordancia con la escritura en español, para este caso pasaría todo el texto a minúsculas. Debe hacerse hincapié en que las correcciones tienen sentido únicamente para el idioma español.
- **GuardaPesos:** Guarda los pesos que se encuentran actualmente en memoria al archivo de pesos del usuario usado en ese momento. Toma todos los pesos y reescribe el archivo de pesos creado en el momento del entrenamiento por la función *nusuario*. Con el objeto de ir guardando los cambios que realiza el usuario en su escritura.

Rutinas para el menú *Ayuda*:

- **acercade:** Muestra un cuadro de mensaje con información acerca del programa y su autor.
- **ayudaP:** Llama a la función principal del módulo de ayuda, la función *vayuda.*

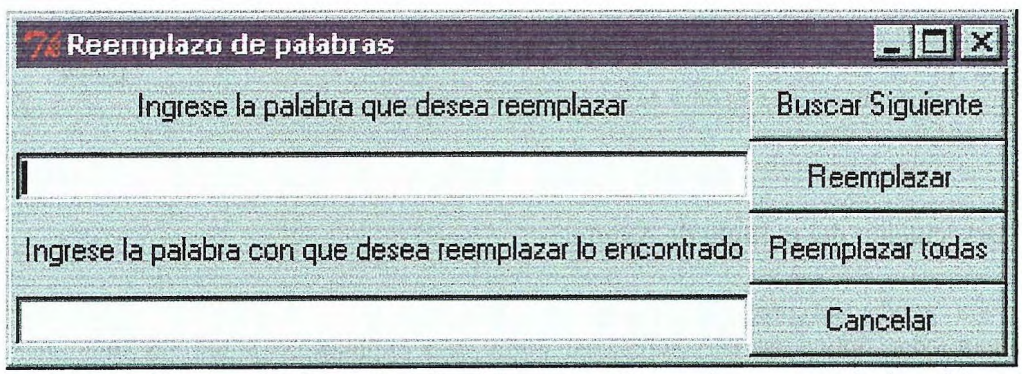


Figura 3.1.2.1.1. Ventana para el reemplazo de palabras, se puede reemplazar una o todas las palabras a la vez.

En la siguiente tabla se muestran los eventos generados por el usuario que provocan que se llame una función del programa:

Evento del usuario:	Función que se llama:
Seleccionar del menú Archivo la opción *Abrir Archivo* Presionar Alt + a. Presionar el botón *Abrir Archivo* de la barra de herramientas.	***AbrirArchivo***
Seleccionar del menú Archivo la opción *Guardar Archivo.* Presionar Alt + g Presionar el botón *Guardar* de la barra de herramientas.	***GuardarArchivo***
Seleccionar del menú Archivo la opción Guardar Como.	***GuardarArchivoComo***
Seleccionar del menú Archivo la opción *Nuevo.* Presionar Alt + n. Presionar el botón *Nuevo* de la barra de herramientas	***NuevoArchivo***
Seleccionar del menú Archivo la opción *Nuevo Usuario.* Presionar Alt + c. Presionar el botón *Nuevo Usuario* de la barra de herramientas.	***nusuario***
Seleccionar del menú Archivo la opción *Usuario.* Presionar Alt + u. Presionar el botón *Usuario* de la barra de herramientas.	***usuario***
Seleccionar del menú Archivo la opción *Guardar Parámetros del usuario.*	***gusuario***
Seleccionar del menú Archivo la opción *Salir.*	***salir***
Seleccionar del menú Edición la opción *Seleccionar Todo.*	***SelTodo***
Seleccionar del menú Edición la opción *Borrar Todo.* Presionar Alt + b.	***BorrarTodo***
Seleccionar del menú Edición la opción *Reemplazar.* Presionar Alt + r Presionar el botón *Reemplazar* de la barra de herramientas.	***Reemplazar***
Seleccionar del menú Procesamiento la opción *Parámetros del segmentador.* Presionar el botón *Parámetros del segmentador* de la barra de herramientas.	***ParSeg***
Seleccionar del menú Procesamiento la opción *Imagen a Procesar.* Presionar Alt + i. Presionar el botón *Imagen a Procesar* de la barra de herramientas.	***AbrirIm***
Seleccionar del menú Procesamiento la opción *Corrección automática*	***Corrige***
Seleccionar del menú Procesamiento la opción *Guardar pesos de la red.*	***GuardaPesos***
Seleccionar del menú Ayuda la opción *Acerca de...*	***Acercade***
Seleccionar del menú Ayuda la opciòn *Ayuda del programa.* Presionar Alt + y. Presionar el botón *Ayuda del programa* de la barra de herramientas.	***AyudaP***

Tabla 3.1.2.1.1. Eventos de usuario que generan una llamada a una rutina de la *IGU.*

3.1.3 Descripción del módulo de Ayuda.

La ayuda del programa ROCM muestra algunos conceptos básicos acerca del reconocimiento óptico de caracteres, así como una explicación sobre las funciones del programa, los parámetros del segmentador, y solución a algunos problemas de reconocimiento.

Ahí aparece el contenido de las imágenes que se usan para la creación de un nuevo usuario y como un usuario ya existente puede acceder a su configuración.

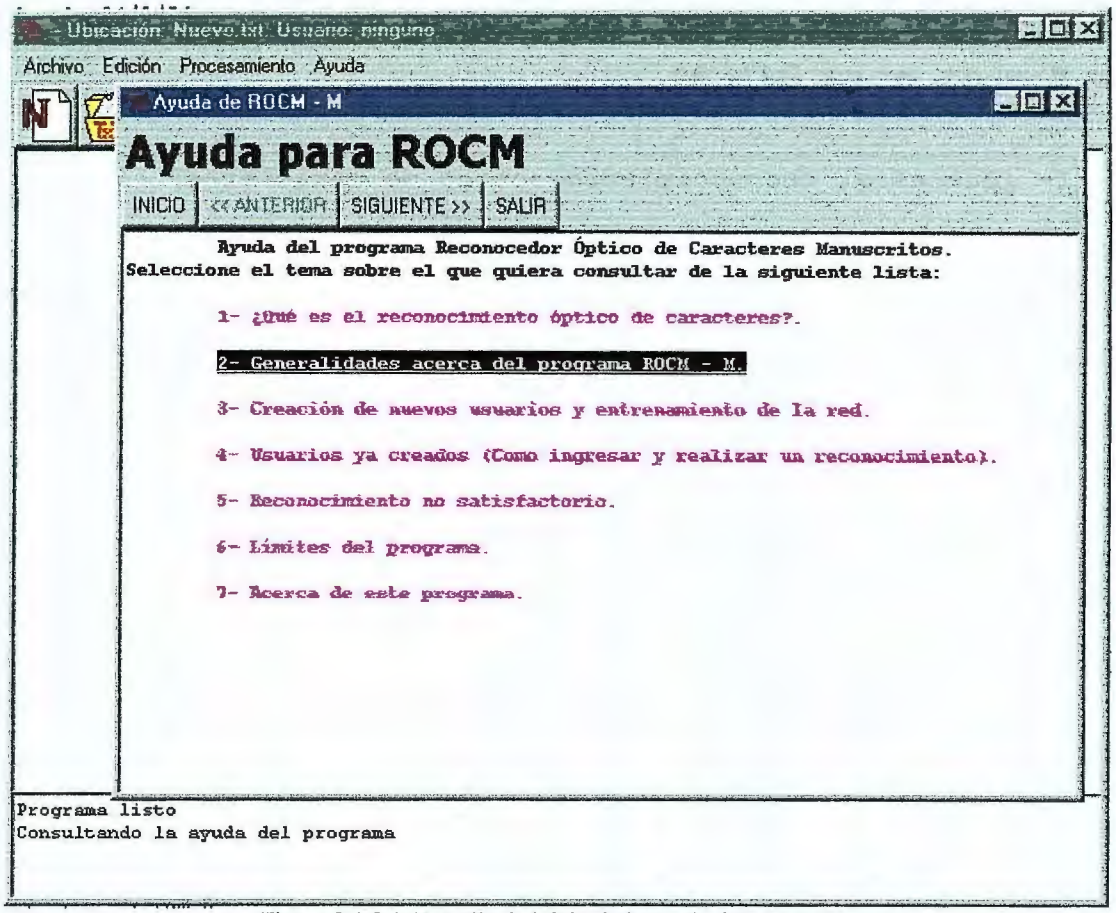


Figura 3.1.3.1. Pantalla de inicio de la ayuda del programa.

La navegación a través de los distintos temas de la ayuda se hace por medio de los botones de desplazamiento que aparecen en la parte superior de la ventana donde se muestra la ayuda o por medio de los vínculos que aparecen en la pantalla de inicio de la ayuda.

Los botones *Anterior* y *Siguiente* estarán habilitados o deshabilitados dependiendo del tema en que se encuentre en ese momento el usuario, por ejemplo si se encuentra en la pantalla de *Inicio*, el botón *Anterior* estará deshabilitado pues no hay un tema anterior a ese al que se pueda acceder, pero el botón *Siguiente* si estará habilitado.

El contenido que se muestra en la ventana de ayuda se encuentra en varios archivos de texto que tienen una extensión "*.ayu*". Dichos archivos se encuentran en la carpeta *ayuda* del programa.

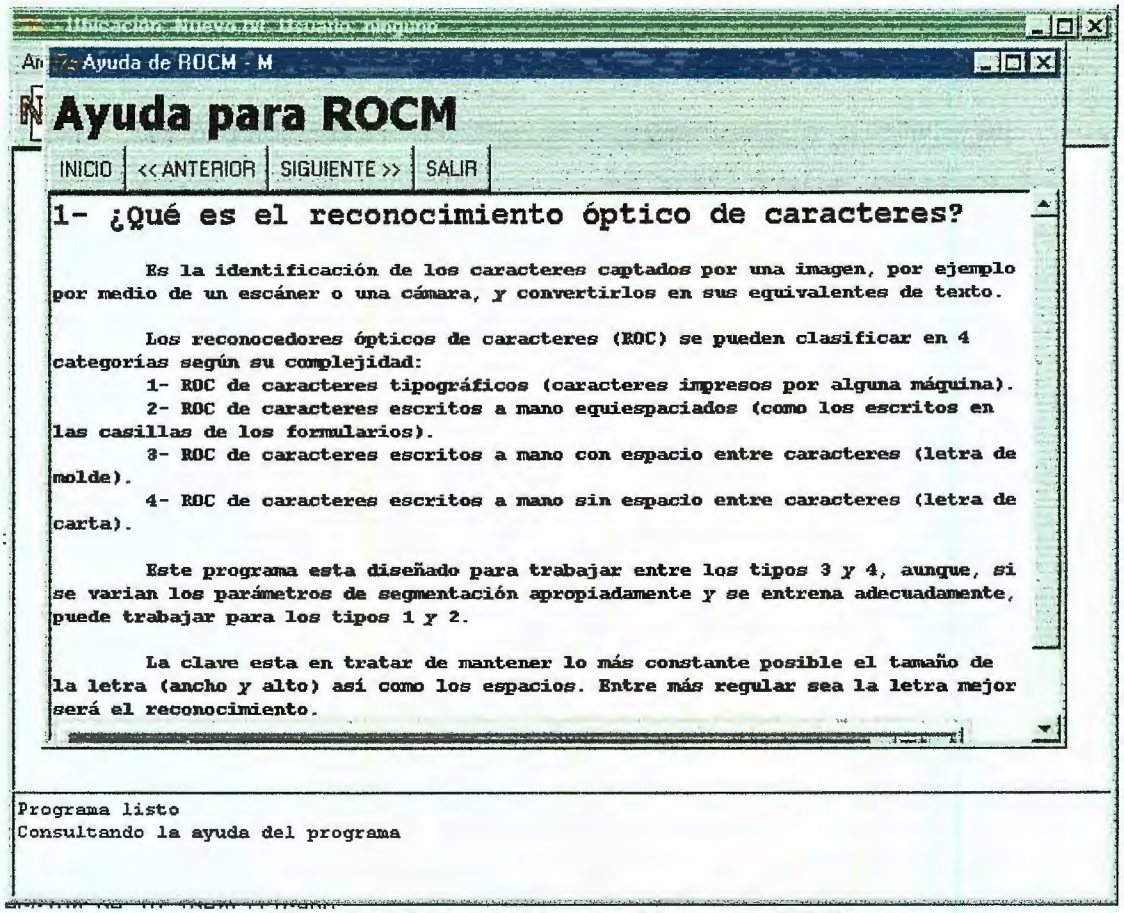


Figura 3.1.3.2. Tema 1 de la ayuda del programa ROCM

3.1.4 Programación del módulo de ayuda.

El diagrama de flujo del módulo de ayuda se muestra en la figura 3.1.4.1. Cuando el módulo es llamado crea su propia ventana donde coloca los controles de navegación y la caja de texto donde aparece el contenido de la ayuda. Se inicializa el valor de la variable *$navegante* a 0, esta variable se encarga de controlar el tema de ayuda que se muestra y el valor de 0 corresponde a mostrar la página de Inicio.

En la página de Inicio aparecen vínculos a los temas de ayuda existentes. Los vínculos se hacen usando texto normal al que se le ha asociado una etiqueta (*tag*) que espera a que se ejecute la acción de pinchar con el ratón sobre ellos. Cuando sucede ese evento entonces varían el valor de la variable *$navegante* al valor del tema escogido, por ejemplo si se pincha sobre el tema 1 entonces *$navegante = 1*, con este valor como argumento se llama a la función *tema* del módulo de ayuda, el cual abre un archivo que tenga de nombre el valor de *$navegante* con extensión *.ayu*, para este caso se formaría "*1.ayu*".

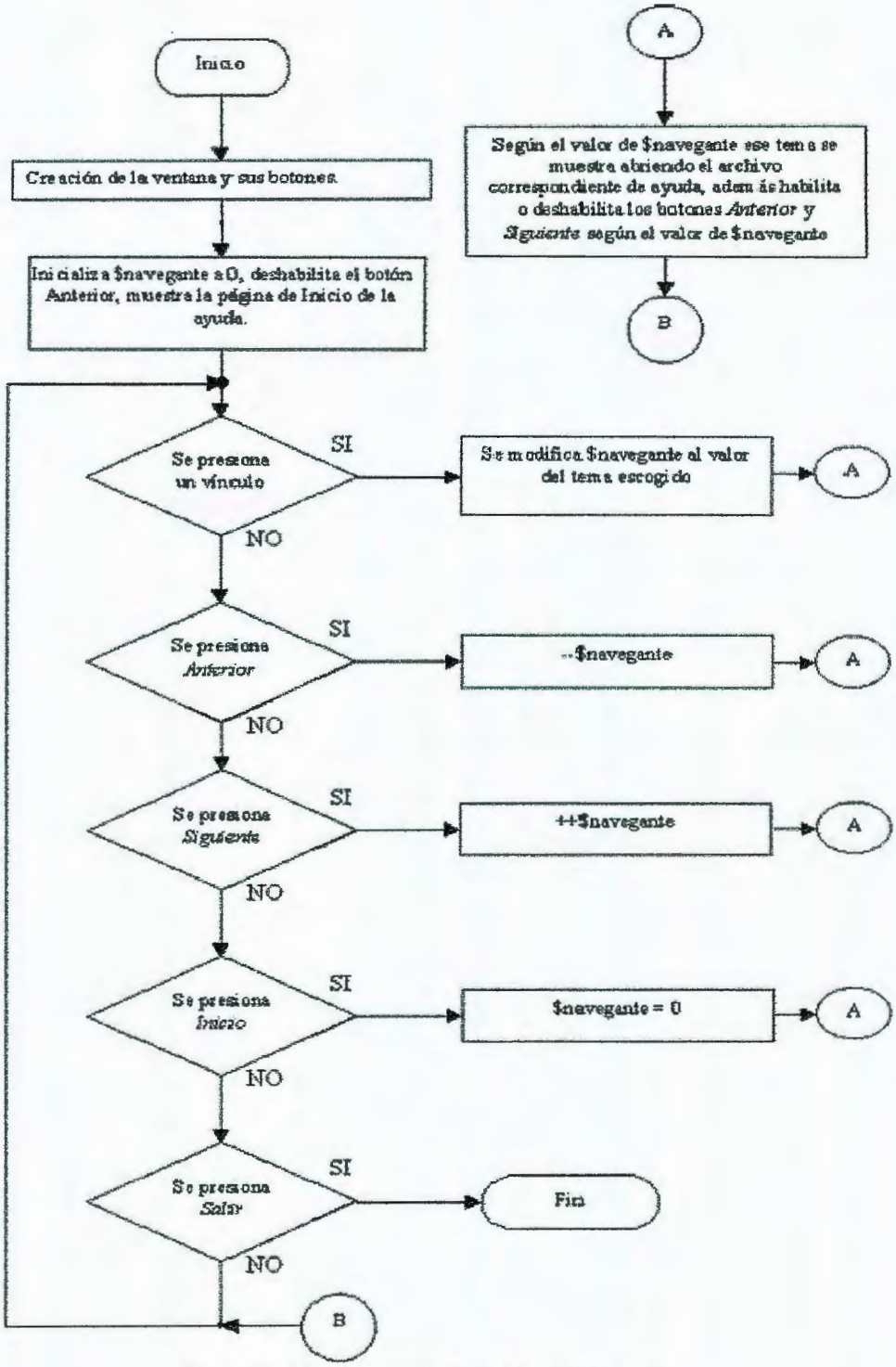


Figura 3.1.4.1. Diagrama de flujo del módulo Ayuda

3.1.4.1 Rutinas ocupadas en el módulo de ayuda

Las rutinas del módulo *ayuda* son:

- **MostrarAyuda:** Crea la ventana de ayuda y los controles de navegación. Llama a la rutina *MostrarInicio*.
- **MostrarInicio:** Establece la variable *$navegante* a cero, deshabilita el botón *anterior* y muestra los temas de la ayuda en la caja de texto principal de la ventana de ayuda. El texto que muestra cada tema es un vínculo creado por medio de etiquetas (*tag*[7]) se establece un formato para que destaque del resto del texto, cuando el usuario pincha con el ratón sobre él se genera un evento que llama a la rutina *tema*, pasando como argumento el número del tema activado. Si el usuario presiona el botón Inicio llama a *MostrarInicio*, si presiona Anterior llama a *Tema* decrementando en 1 a *$navegante*, si presiona Siguiente llama a *Tema* aumentando en 1 a *$navegante*.
- **Tema:** Si el argumento vale 0 llama a *MostrarInicio*, para todos los otros valores abre el archivo que se forma con *argumento* y la extensión *.ayu*, por ejemplo para el tema 1 llamaría a *"1.ayu"*, el archivo es buscado en la carpeta ayuda del directorio de trabajo. Si *$navegante* es mayor que 0 habilita el botón *Anterior*, de lo contrario lo deshabilita. Si *$navegante* es menor que 6 habilita el botón *Siguiente*, de lo contrario lo deshabilita. También se encarga de llamar a la función *InIm*, en caso de que el tema que se este consultando posea imágenes.
- **InIm:** Esta función es llamada por la función *Tema*, y se encarga de insertar las imágenes en los temas que las poseen. Necesita como argumento el nombre del archivo de imagen a insertar además de la posición en donde será insertada.

3.1.5 Descripción del módulo que maneja el asistente para crear nuevos usuarios.

El programa reconocedor es capaz de crear usuarios para personalizar los parámetros del segmentador y, para la defensa final, de la red neuronal para ajustarse a la escritura del usuario.

Para crear nuevos usuarios se accede al asistente de Creación de nuevos usuarios, por medio de *Archivo→Nuevo usuario*, presionando *Alt + c*, o por medio del botón de la barra de herramientas correspondiente. La primera pantalla que aparece se muestra en la figura 3.1.5.1. A medida que se vaya desarrollando la creación del usuario se pedirán varias imágenes que contenga ciertas letras[8], las cuales servirán de patrón al programa para establecer los parámetros de segmentación, en primer lugar, y, posteriormente, los pesos de la red neuronal para identificar la escritura del usuario.

En los pasos de modificación de los pesos se pedirá que en las imágenes se encuentren todos los caracteres que el reconocedor es capaz de manejar[9].

---

[7] *tag* con este nombre se distingue una porción de texto seleccionada a la que se le asigna un nombre y formato propios.

[8] Ver Anexos A.1 Ayuda del programa, ahí se muestran todos los caracteres que es capaz de reconocer, así como el contenido que deben tener las imágenes.

[9] Ver Anexos A.1 Ayuda del programa, ahí se muestran todos los caracteres que es capaz de reconocer, así como el contenido que deben tener las imágenes.

El programa de reconocimiento no se encarga de digitalizar imágenes, por lo cual dichas imágenes deberán obtenerse por otro programa o aplicación. Es importante que los patrones usados en el entrenamiento se asemejen a la escritura del usuario y que contengan exactamente los caracteres que se piden, de lo contrario no se asegura un buen reconocimiento.

Al final de la creación se le pedirá al usuario un nombre para identificarlo, aparte de servir para la creación de los archivos donde se guardarán los parámetros de su configuración.

Una vez creado el usuario se podrá acceder a su configuración ingresando con el nombre con el que se creo el usuario como se muestra en la ayuda del programa[10] y posteriormente accediendo al menú Procesamiento para consultar los parámetros del segmentador.

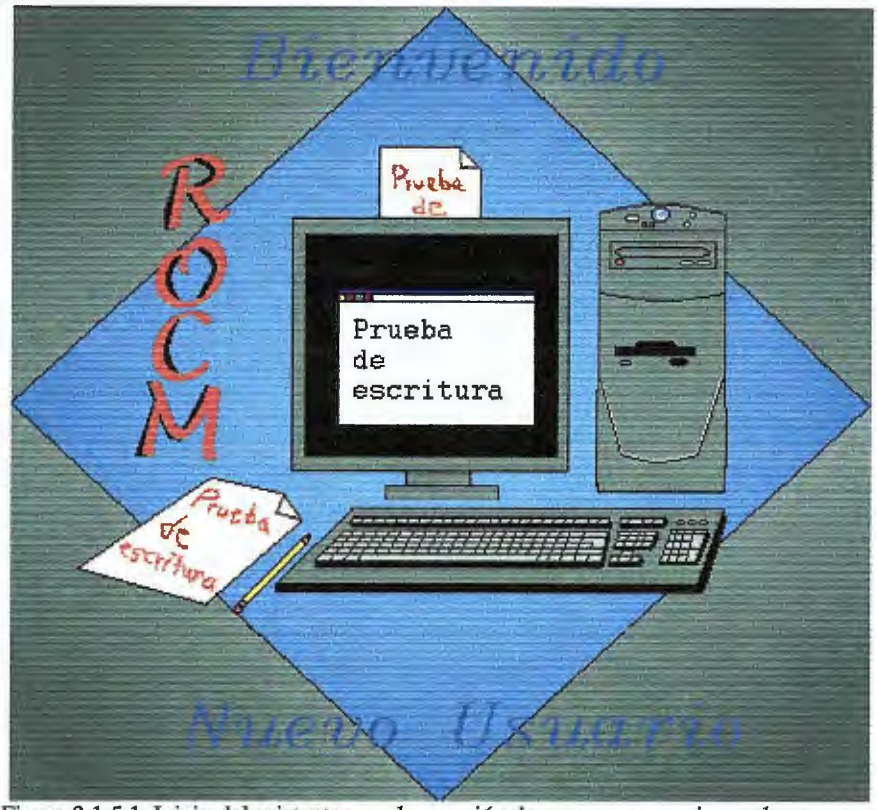


Figura 3.1.5.1. Inicio del asistente para la creación de un nuevo usuario en el programa.

Son en total 5 pasos los que tiene esta rutina para modificar los parámetros del segmentador y también se encarga de modificar los valores de la red art. En cada paso realizado durante el entrenamiento se muestra el resultado de la segmentación realizada en la imagen que se le ha ingresado, con esto el usuario puede modificar los parámetros de tamaño de ventana en x y y, además de las tolerancias, a fin de mejorar la adaptación del programa a la escritura del usuario. Es decir que aparte de que el programa verifica que se encuentre el número de caracteres y líneas que se buscan en cada paso el usuario da el visto

[10] Ver Anexo A.1.4.

bueno si le parece que las imágenes segmentadas representan de una forma adecuada lo que él ha ingresado.

3.1.6 Programación del módulo que maneja el asistente para crear nuevos usuarios.

Para el asistente para crear nuevos usuarios se tienen los siguientes diagramas de flujo:

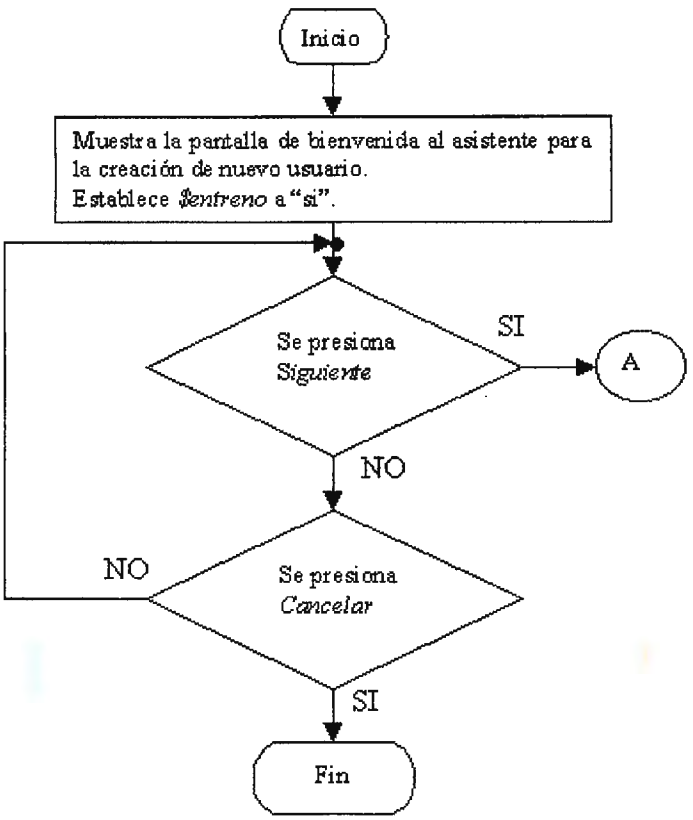


Figura 3.1.6.1. Inicio del asistente para crear nuevos usuarios.

Se crea la ventana de inicio para el asistente de crear nuevo usuario, aparecen dos botones (*siguiente* y *cancelar*), si se presiona cancelar en cualquiera de los pasos siguientes entonces finaliza el asistente y no crea ningún usuario.

Se establece el valor de la variable *$entreno* en "si", con esto al llamar al segmentador éste sabe que se esta realizando una creación de un usuario por tanto le pasa el control, la llamada de las siguientes funciones, al módulo *nusuario*, de esta manera el módulo puede llamar cualquier función del segmentador las veces que sea necesaria.

Es importante hace notar que si se cancela la creación o se cierra la ventana del asistente la variable *$entreno* toma el valor de "no", para que el segmentador funcione normalmente.

Al presionar el botón de siguiente se llama a la función *paso1*, la cual se encarga de variar los parámetros de *$tx*, *$ty* *$tolx* y *$toly*, según la imagen ingresada por el usuario, la cual esta conformada por 3 líneas y 17 letras ('Prueba de escritura'), sin tomar en cuenta los espacios.

Como se conoce de antemano lo que debe encontrarse en la imagen entonces se realiza un lazo, variando las tolerancias y el tamaño de la ventana en cada vuelta ya sea aumentándolo o lo contrario, de manera que se alcance el valor esperado.

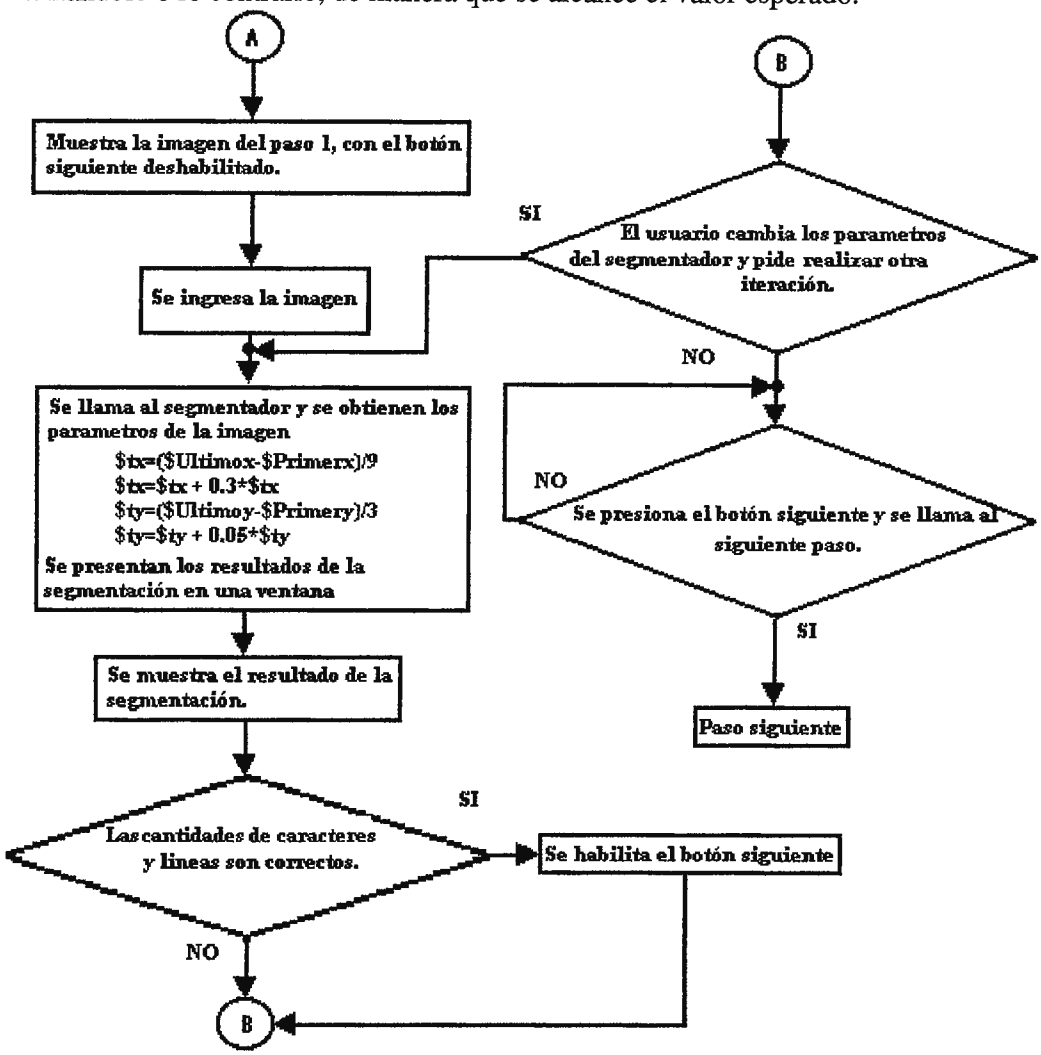


Figura 3.1.6.2. Paso 1 del asistente para crear nuevos usuarios. El lazo de llamada a la función recortar se realiza hasta encontrar el número de letras y líneas esperado.

En la figura 3.1.6.2 se puede apreciar que el valor de *$tx* y *$ty* depende del tamaño de la imagen que se encuentre. El tamaño de la imagen se conoce gracias a la función *CreaMatriz* del módulo recortar la cual regresa la primera y última posición en *x* donde se encuentra un pixel en negro y también la primera y última posición en *y* de un pixel en negro. Con estos datos se forma un 'marco" que sirve de límite para la función *recortar* del módulo *segmentar* y también sirve para establecer los valores de *$tx* y *$ty*, de la siguiente manera:

$$\$tx = (UltimoX - Primero X)/9 \quad \text{Ecuación 3.1.6.1.}$$

$$\$tx = \$tx + 0.3 * \$tx \quad \text{Ecuación 3.1.6.2.}$$

$$\$ty = (UltimoY - Primero Y)/3 \quad \text{Ecuación 3.1.6.3}$$

$$\$ty = \$ty + 0.05 * \$ty \quad \text{Ecuación 3.1.6.4}$$

Los valores de *$tolx* y *$toly* dependen de *$tx* y *$ty*, así *$tolx* es el 10% de *$tx* (redondeado a el valor entero más próximo superior a 1) y *$toly* es igual al 20% de *$ty* (redondeado al entero más próximo superior a 1) . De las ecuaciones anteriores se puede ver que *$tx* aumenta en un 30% y *$ty* en un 5%, esto se hace para tener un margen de error, es mejor que la ventana sea un poco más grande que lo necesario, siempre y cuando no sea tan grande como para provocar una mala segmentación. Los valores de aumento han sido establecidos en base a prueba y error, usando diferentes tipos de letras.

El botón de siguiente del paso 1 esta deshabilitado, hasta que se encuentra el número de letras y líneas correcto, sino se encuentra entonces ese botón sigue deshabilitado y no se puede proseguir con el entrenamiento.

Los pasos del 1 al 5 son llamados usando la función *pasoA* del módulo *nusuario*, esta función se encarga de ir aumentando el valor de la variable *$paso* que representa el número del paso en que se encuentra el proceso. Dependiendo del paso en que se encuentre el proceso tendrá un número de letras y líneas que cumplir, esta variable *$paso* también es consultada por otras funciones para saber cual es la acción que deben tomar, por ejemplo la función *entrenar* se encarga de realizar el preprocesamiento de la imagen y entrenar a la red pero solo si se encuentra con que el valor de *$paso* es $2 \leq \$paso \leq 4$.

Los pasos siguientes siguen la misma lógica del paso 1, presentan la imagen con las letras que se buscan, se abre un archivo de imagen para ser segmentado, con la única diferencia de que al terminar la segmentación se muestra una ventana donde aparecen los caracteres como han sido segmentados, si se encuentra las cantidades de letras y líneas correctos habilita el botón para aceptar el resultado de la segmentación, al presionar dicho botón llama a la función *entrenar*, que se encarga de entrenar la red con los resultados de la segmentación y realiza la relación entre la salida activada en la red con la letra que se busca actualmente. Esto último se hace de la siguiente manera: si por ejemplo el entrenamiento se encuentra en el paso 2 y se ingresa una imagen donde se encuentran las mayúsculas, la primera mayúscula que debe estar presente en la imagen es la letra 'A', por tanto el primer resultado de la segmentación debe ser la correspondiente a dicha letra, por tanto cualquiera que sea la salida que se active será "relacionada" con la letra 'A'. Así cuando se realice el reconocimiento y se active esa salida se toma que ha reconocido la letra 'A'. Lo mismo sucede para todos los caracteres que es capaz de reconocer el programa. Cuando ha terminado de entrenar la red con los caracteres correspondientes a ese paso entonces muestra la imagen del siguiente paso, se busca el número de líneas y caracteres correcto y se vuelve a entrenar la red con esa nueva información, y así continua hasta llegar al paso, donde al presionar el botón finalizar muestra el cuadro de diálogo para ingresar el nombre del usuario.

Figura 3.1.6.3. Cuadro de diálogo para ingresar el nombre para identificar el usuario recién creado.

Al presionar el botón aceptar, o presionando cancelar en cualquiera de los pasos, se da por finalizado la creación del nuevo usuario, en el primer caso dando como resultado varios archivos que contienen los parámetros del segmentador, lo pesos de la red y la relación de las salidas de la red con los caracteres reconocidos por el OCR. Cada uno de esos archivos posee una extensión distinta para identificarlos y su nombre es igual al nombre del usuario así los tres archivos formados serían:

- *$usuario.par* Contiene los parámetros del segmentador.
- *$usuario.pes* Pesos de la red.
- *$usuario.rel* La relación entre las salidas del segmentador y los caracteres.

Todos los archivos están en formato de texto.

3.1.6.1 Rutinas ocupadas en NUSUARIO

El objetivo de *nusuario* es el de entrenar a la red neuronal y además encontrar los parámetros adecuados para el segmentador. Para ello ocupa las siguientes funciones:

- **inicia:** Se crea la ventana principal del asistente de creación de usuarios, se establece la variable *$entreno* a "si" para indicarle al segmentador que se esta creando un usuario. Si se cierra la ventana o se presiona el botón cancelar, entonces cambia el valor de *$entreno* a "no". Además establece el orden de los caracteres que serán reconocidos y asigna que la primera letra a reconocer es la 'A', esto gracias a la variable *$posCar*, que se encarga de controlar cual letra es con la que esta trabajando. Al presionar el botón *siguiente* llama a la rutina *pasoA*.
- **pasoA:** Se encarga de controlar el valor de la variable *$paso* para saber en que parte del proceso se encuentra el programa, dependiendo de su valor va estableciendo las condiciones para poder pasar a la siguiente etapa. En cada paso muestra la imagen que contiene los caracteres a encontrar, y al presionar en el botón *abrir imagen* llama a la función que se encuentra en el módulo *IGU abrirIm*, la cual se encarga de presentar el cuadro de diálogo para que el usuario pueda seleccionar la imagen a procesar. En este punto el módulo *nusuario* le pasa el control a la función *mostrarI,* que presenta la imagen seleccionada junto con el botón *procesar*, al ser presionado invoca a *procesar* que se encuentra en *nusuario* para continuar el proceso. Lo anterior es válido hasta llegar al paso 5 pues en este caso ya no presenta la opción de abrir una imagen sino que presenta al usuario un cuadro de diálogo donde escriba su nombre que le servirá como identificador, se graban tres archivos el primero se forma así: *nombre del usuario.par* donde se encontrarán los parámetros del segmentador generados por la etapa de creación de nuevo usuario, el segundo archivo es *nombre del usuario.pes* que contiene los pesos de la red, el último es *nombre del usuario.rel* que contiene la relación entre las salidas de la red neuronal y las letras que puede reconocer el OCR, además finaliza el módulo *nusuario.* Si en cualquier momento se presiona el botón *Cancelar* el proceso

termina estableciendo el valor de *$entreno* a 'no', se destruye la ventana del paso en que se encuentre y pasa el enfoque a la ventana principal. En el primer paso se establece la variable *$n*, la cual contiene la cantidad de datos que se usarán para establecer el total de entradas del segmentador, y por tanto el número de datos que debe devolver la función *tdc* del módulo *vecino8* para que a cada entrada de la red llegue un dato. Es en este momento cuando se inicializa la red neuronal con el número de entradas dadas por *$n*, llamando a la función *art2Incio* pasándole *$n* como argumento, los demás parámetros de la red son leídos desde el archivo *configuraart2.ini*. Al final del paso 5, una vez que el usuario ha ingresado un nombre, esta función invoca a *art2_guardar* para grabar los pesos creados durante el entrenamiento, con el nombre dado por el usuario, el cual es pasado como argumento al llamar esta función.

- **procesar:** Al ser presionado el botón *procesar* que aparece junto a la imagen a trabajar, esta rutina toma el control y llama a las funciones del módulo *segmentar* según sea requerido. El objetivo de procesar es encontrar la cantidad de letras y líneas establecidas para cada paso, que se esperan estén en la imagen provista por el usuario, se realiza un lazo hasta tener el número de letras y líneas mostrando en cada iteración el resultado de la segmentación este lazo puede terminar presionando el botón cancelar. Cuando se han establecido las metas del proceso se llama a la función *lazo* del módulo *nusuario* el cual se encarga de llamar a *CreaMatriz* que se encarga de buscar los límites del *marco* que contiene la imagen. En el paso 1 con los valores regresados por *CreaMatriz* se establecen *$tx* y *$ty*, además de las tolerancias, en los otros pasos no se realiza esta asignación de valores sino que estas variables dependen de las variaciones que realice el usuario en esos parámetros. Se llama a la función *lazo* quien a su vez llama a *recortar* del módulo *segmentar*, dicha función regresa el número de letras y líneas encontradas en la imagen y necesita como parámetros los límites del *marco* de la imagen, los valores de *$tx, $ty, $tolx* y *$toly*. En cada llamada a la función recortar se muestra el resultado de la segmentación llamando a la función *muestra*, que muestra los caracteres de la manera en que han sido segmentados por el programa, de esta manera el usuario puede verificar si se ha realizado de manera satisfactoria, o si desea variar los parámetros del segmentador, pero ninguno de estos parámetros puede tener un valor menor que 1. Cuando la cantidad de letras y líneas es la esperada se pasa a la próxima etapa que habilita el botón *siguiente* y no se podrá ir al siguiente paso a menos que se ingrese una imagen que cumpla con los requisitos de letras y líneas esperados.
- **lazo:** Es llamada en primera instancia por la función *procesar* para iniciar el proceso de la segmentación de la imagen, pero también puede ser llamada por la función *muestra* si el usuario decide hacer otra segmentación. Cuando es invocada llama a *recortar* del módulo *segmentar* el cual a su vez devuelve el número de líneas y letras encontrados Con estos datos varía a verdadero las variables *$CumpleC* y *$CumpleL* si se han encontrados las cantidades de caracteres y líneas correctas respectivamente o falso si no es cierto. La función *muestra* es invocada después de cada llamada a *recortar*. Dependiendo de los resultados regresados por *recortar* muestra algunas recomendaciones a los usuarios sobre la variación de lo parámetros para llegar al resultado correcto.
- **muestra:** Es invocada por *lazo*. El resultado de una segmentación siempre se guarda en un archivo temporal que es accesado por esta función, el cual contiene las letras de la manera en que el programa las ha segmentado, es importante saber si lo ha hecho

correctamente pues este resultado será usado para entrenar a la red y si no es correcto no se pueden esperar reconocimientos acertados. Para poder representar lo contenido en el archivo temporal resultante de la segmentación esta función traduce los valores allí encontrados en los términos 'black' si el valor es 1 y ' white' si el valor es 0, dichos términos son comprendidos por el módulo TK y los traduce en cambios en el color de los pixeles de la ventana *muestra*. Esta función verifica si se ha cumplido o no con las condiciones para continuar con el proceso consultando con las variables *$CumpleC* y *$CumpleL* las cuales son controladas por la función *lazo*. Si alguna de dichas variables es falsa entonces no habilita el botón de *Aceptar el resultado*. Cuando ambas variables poseen el valor 'verdadero' el botón puede ser presionado y al hacerlo llama a la función *entrena* perteneciente a *nusuario*, para continuar con el proceso. En la figura 3.1.6.1.1. aparece la imagen resultante de una segmentación durante el entrenamiento, se puede apreciar que posee botones para incrementar y decrementar los valores de los tamaños de la ventana del segmentador y las tolerancias, ninguno de esos valores puede valer menos de 1, y si se ingresa algo menor es cambiado a 1. Si se ingresa un número decimal solo se toma la parte entera. Para realizar los decrementos e incrementos de los valores usando los botones de comando se llama a las funciones: *Txd, Txa, Tyd, Tya, Tolxd, Tolxa, Tolyd* y *Tolya*.

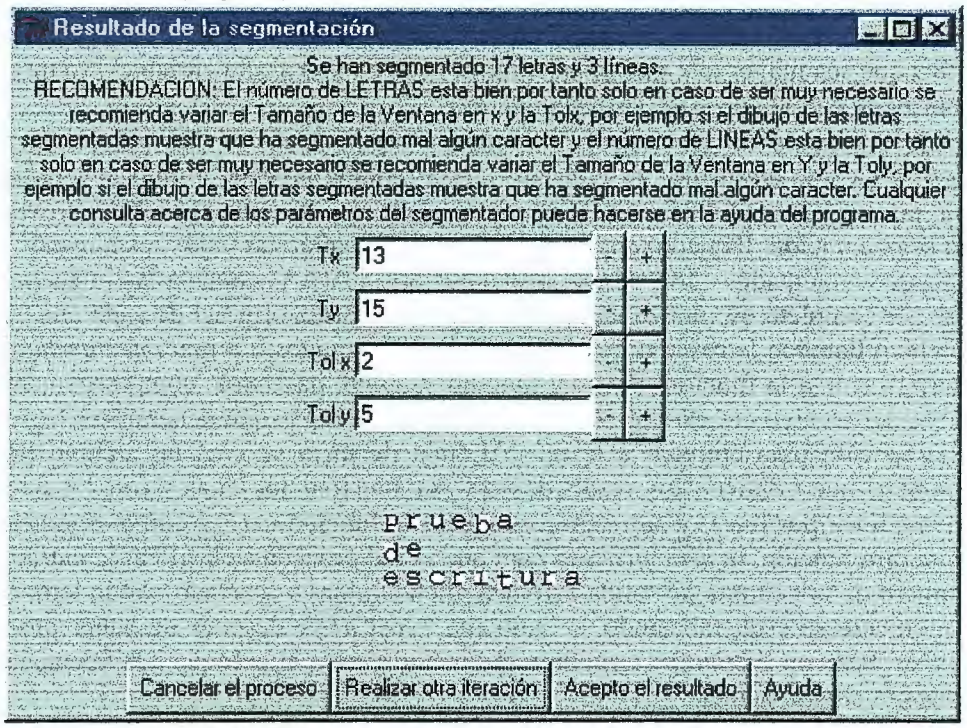


Figura 3.1.6.1.1. Cuadro de diálogo creado por la función *muestra* donde aparece el resultado de una segmentación, además muestra los valores de los parámetros del segmentador los cuales pueden ser alterados por el usuario, y recomendaciones para alterar los valores del segmentador.

- **Txd:** Toma el valor existente en *$tx* lo decrementa en uno, toma su parte entera, verifica que sea mayor que 1 de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.

- **Txa:** Toma el valor existente en *$tx* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tyd:** Toma el valor existente en *$ty* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tya:** Toma el valor existente en *$ty* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolxd:** Toma el valor existente en *$tolx* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolxa:** Toma el valor existente en *$tolx* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolyd:** Toma el valor existente en *$toly* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolya:** Toma el valor existente en *$toly* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **entrenar:** Cuando se ha conseguido que el segmentador divida correctamente la imagen y ha sido aprobado por el usuario, esta función procede a invocar a la función *zoom*, perteneciente al módulo del mismo nombre, el cual toma el archivo proveniente de la segmentación, llamado *letras.txt*, y procede a regularizar todos los tamaños de las imágenes segmentadas a los valores dados por *$tx* y *$ty*, y aumenta las imágenes para llenar por completo ese nuevo tamaño sin perder la forma original de cada una, se crea un archivo llamado *letras2.txt* que contiene estos resultados. La función *entrenar* llama posteriormente a la función *tdcm* del módulo *vecino8*, pasándole como argumentos, el nombre del archivo creado por *zoom* y la cantidad de datos que deberán ser regresados por la función para entrenar a la red. Al final de *tdcm* se crea un archivo llamado *destino.txt*, que contiene los resultados de esa función, los cuales son tomados por *entrenar* línea por línea, y son suministrados a la función *art2_1* como argumento, se ejecuta otra función, la *art2_procesa* para que la red proceda a clasificar la entrada y da el número de la neurona de salida activada, dicho valor es asociado con la letra que en ese momento se esta trabajando, se aumenta *$posCar*, y después de haber procesado todos los datos en *destino.txt* termina esta función y pasa el enfoque a la ventana donde se muestra el paso actual.

3.1.7 Descripción del módulo para la corrección automática.

Cuando se ha realizado una identificación hay errores que son bastante comunes, como por ejemplo que aparezcan letras mayúsculas y minúsculas mezcladas, para este caso se podría tener que todas las letras deberían ser mayúsculas, caso menos probable, o que todas sean minúsculas, caso más probable, excepto por el primer caracter que podría ser el

inicio de un nombre propio, por tanto lo más conveniente sería cambiar todas las letras mayúsculas a minúsculas, pero si la letra inicial es mayúscula esa no se vería afectada por este cambio. Otro problema bastante común es el poner el número uno en lugar de la letra 'l', por tanto si lo que antecede o esta posterior al caracter 1 son letras entonces es cambiado por 'l' y viceversa.

El módulo encargado de realizar estos cambios es *reglas*. El cual es invocado por la IGU, se puede acceder a la corrección automática a través del menú *Procesamiento*, o presionado el botón de la barra de herramientas correspondiente. En la tabla 3.1.7.1 aparecen todas las correcciones que actualmente realiza este módulo.

Si aparece:	Es reemplazado por:
Un uno antes de una letra.	Letra 'l'.
Punto, un espacio y una letra minúscula.	Cambia la letra a mayúscula
Letra mayúsculas después de letras minúsculas.	Letras minúsculas.
Una 'ñ' al final de una palabra.	Letra 'n'.
Una 'c' al principio de una palabra y la letra siguiente no es 'r', ni 'l', ni vocal.	Letra 'e'.
'Gos'.	'Cos'.
Si aparece un uno justo después de una letra.	Letra 'l'.
La letra 'q' antes de cualquier letra que no sea 'u'.	Letra 'g'.
Letra 's' entre dos consonantes.	Letra 'a'.
Letra 'g' al final de una palabra.	Letra 'a'.
El signo '¡' entre dos letras.	Letra 'i'.
Un '9' antes de una 'u'.	Letra 'q'.
Un '9' antes de cualquier letra excepto 'u'.	Letra 'g'.
Un '$' antes o después de cualquier letra.	Letra 's'.

Tabla 3.1.7.1 Correcciones hechas por el módulo *reglas*. Se puede activar después del reconocimiento para mejorar la identificación de los caracteres.

Las correcciones hechas por éste módulo solo tienen sentido para el idioma español. Además están hechas en base de que lo escrito tiene sentido y no son caracteres al azar sino palabras con significado. La aplicación de este módulo es opcional pudiendo el usuario escoger en no aplicarlo.

3.1.8 Programación del módulo *reglas*.

Este módulo ocupa una única función llamada *corrige*, que no necesita argumentos. Cuando es invocada procede a obtener todo el texto contenido en el control de texto principal y lo guarda en la variable *$hilera*, dicha variable es afectada por lo que en Perl se conoce como "expresiones regulares", que es una característica especial de éste lenguaje para procesar información de texto[11]. Gracias a esto Perl puede realizar búsquedas en el texto por un patrón cualquiera, pudiendo ser reemplazado posteriormente. Como el módulo

[11] Ver referencias 7 y 8.

reglas esta implementado vía las expresiones regulares se tiene una gran facilidad para definir los patrones a encontrar y los reemplazos que se desean, ya que hay caracteres comodines que equivalen a todas las letras, solo números, letras y números.

3.1.8.1 Funciones ocupadas por el módulo reglas.

La única función que se ocupa es la siguiente:

- **corrige:** Cuando esta función es invocada por la IGU toma todo el texto que aparece en la ventana de texto, y lo almacena en la variable *$hilera*, que contiene además de los caracteres los cambios de línea y espacios. Posteriormente se aplican una serie de expresiones regulares para encontrar todos los patrones presentados en la tabla 3.1.7.1 junto con los textos de reemplazo, la variable *$hilera* es afectada por estos reemplazos, después se llama a la función *borrarTodo* de la IGU, con lo que se borra todo el texto en la pantalla, y se inserta el contenido de la variable *$hilera*.

3.2 EL SEGMENTADOR

3.2.1 Descripción del segmentador

El segmentador puede ser llamador desde el menú del programa: Segmentación, o desde la barra de herramientas, Parámetros del segmentador y Abrir archivo de imagen como se muestran en las figuras 3.2.1 y 3.2.2 respectivamente.

Los parámetros del segmentador pueden ser ajustados manualmente para acoplarse mejor a la escritura que se quiere identificar.

Un detalle a notar es el hecho de que el programa solo trabaja con archivos de imágenes bmp o gif, además las imágenes deben ser binarias, es decir, solo deben poseer dos colores el blanco (el fondo) y el negro (el texto). De lo contrario no se asegura un correcto reconocimiento.

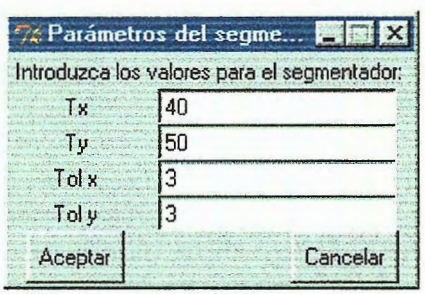


Figura 3.2.1. Ajuste de los parámetros del segmentador.

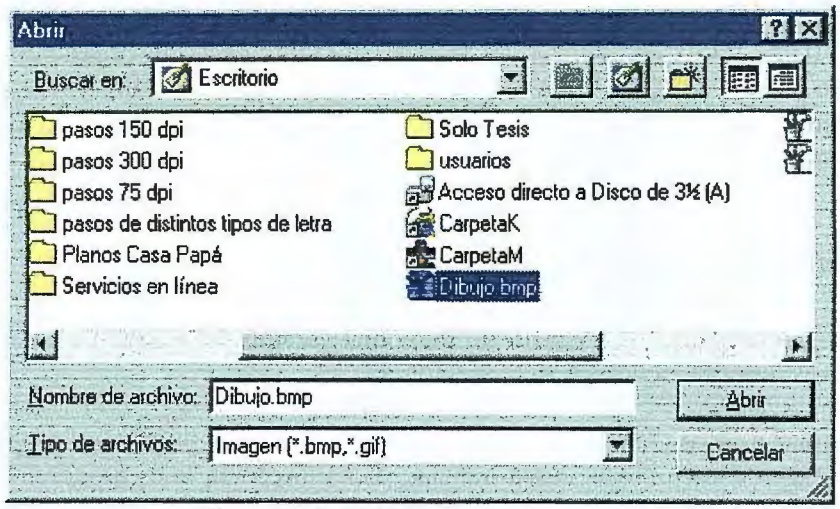


Figura 3.2.2. Abrir Imagen para procesarla.

El segmentador es la parte del programa que se encarga de tomar la imagen y dividirla en regiones cuyo tamaño no sobrepase lo dado por los parámetros Tx (ancho máximo del caracter en pixeles, su valor esta guardado en la variable *$tx*), Ty (altura máxima del caracter en pixeles, almacenado en la variable *$ty*) y Tolerancia (que tan lejos puede estar un pixel y aún se toma como perteneciente a un caracter, la distancia también esta dada en pixeles, valor almacenado en *$tol*). En dichas regiones es donde se presume hay un caracter, y lo primero que hace es encontrar es la primera y última fila con información (pixeles en negro) y la primera y última columna con información (pixeles en negro), después de eso explora de arriba hacia abajo de izquierda a derecha hasta encontrar un pixel en negro desde ese momento comienza a crear una pequeña matriz, llamada ventana, en la cual se ponen todos los pixeles que aparecen conectados al primer pixel o a una distancia de separación máxima dada por el parámetro de Tolerancia, cuando ya se ha alcanzado los valores de Ty y Tx imprime el caracter encontrado en un archivo de texto, con unos y ceros, y se pasa a buscar otro pixel y se repite el proceso hasta llegar al final de la imagen.

Figura 3.2.3. Se ha abierto una imagen que esta lista para ser procesada (segmentada).

Para saber si un pixel esta conectado se evalúa si dicho pixel esta dentro del rango de tolerancia dado ($tolx en la dirección horizontal y $toly en la dirección vertical), yendo un número de pixeles desde la izquierda hacia la derecha, se baja un pixel y se prueba otra vez de izquierda a derecha y se vuelve a bajar, se repite el proceso hasta encontrar un pixel o alcanzar los valores límites del programa.

3.2.2 Programación del segmentador

En la figura 3.2.2.1 se muestra un diagrama general del segmentador, cada rutina que se utiliza es descrita en detalle en la sección 3.2.2.1.

Al llamar a la primera función del módulo *segmentar*, se le debe proporcionar el archivo de imagen con el que se desea trabajar, dicha dirección le es proporcionada por la IGU, por medio de la función *AbrirIm*, cuando se esta trabajando con el asistente para crear nuevos usuarios también hacer una llamada a esa función para que presente el cuadro de diálogo para abrir una imagen. La única diferencia entre el uso del módulo *segmentar* cuando se usa para crear nuevos usuarios y su uso normal es el hecho de que la función *procesar* que activan es distinta, para el primer caso esta función se encuentra en *nusuario* y llama repetidamente a la función *recortar* cuando el usuario decide procesar la imagen y después en cada nueva iteración, para el otro caso la función *procesar* se encuentra en el interior del mismo módulo *segmentar* y solo llama una vez a la función recortar tal como lo muestra la figura 3.2.2.1.

En la misma ventana donde se presenta la imagen aparece el botón de comando *procesar* que, al ser presionado, llama a la función *procesar* ya sea del módulo *segmentar* o del módulo *nusuario*[12]. En ambos casos se llama posteriormente a la función *CreaMatriz* que se encarga de tomar los datos de imagen y ponerlos en una matriz, además esta función llama a cuatro funciones más las cuales se encargan de encontrar la primera y última posición en *x* donde se encuentra un pixel (funciones *PrimerP_x* y *UltimoP_x*), y la primera y última posición en *y* donde se encuentra un pixel (funciones *PrimerP_y* y *UltimoP_y*).

Con esas cuatro posiciones se forma el "marco" que contiene toda la información de la imagen como lo muestra la figura 3.2.2.2.

Dicho "marco" servirá como límite para la función *recortar*, ya que se le debe suministrar el espacio de la imagen donde se encuentra la información que se desea segmentar. Se crea estos límites para que el segmentador no pierda el tiempo con partes en las que definitivamente no hay pixeles en negro.

En el caso de que toda la imagen este en blanco, es decir, no hay pixeles en negro, la función *CreaMatriz* devuelve el valor "verdadero" con lo que la función *procesar* terminaría el llamado al segmentador y devolvería el mensaje "¡La imagen esta en blanco!" en la caja de texto secundaria de la IGU, si la imagen posee pixeles en negro entonces

[12] Para una mejor explicación de la función *procesar* del módulo *nusuario* ver la sección 3.1.6.1.

regresa el valor "falso", que le indica a la función *procesar* que si hay imagen para continuar con la segmentación.

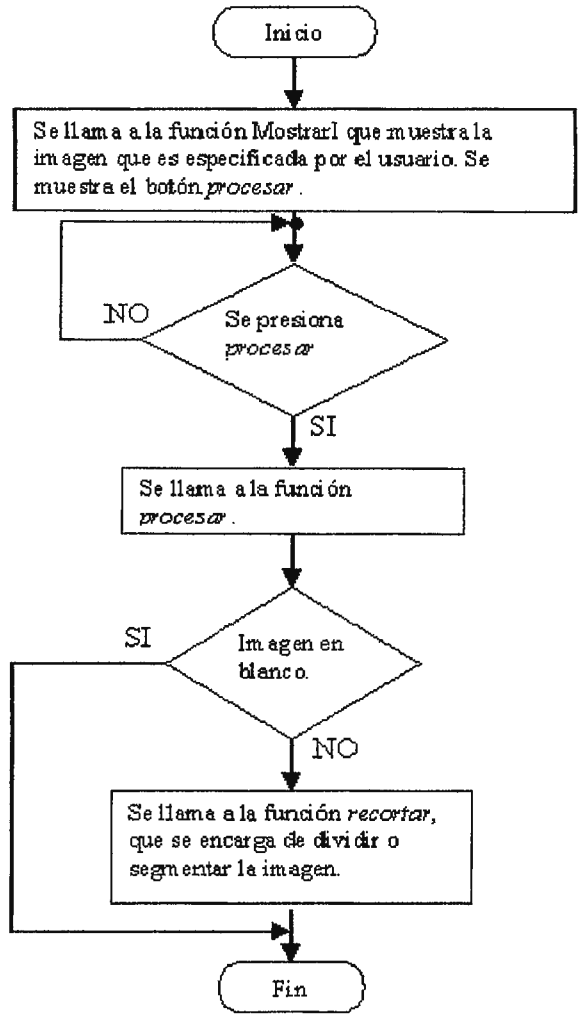


Figura 3.2.2.1. Diagrama de flujo del módulo segmentar.

Cuando existe información en la imagen proporcionada por el usuario la función *procesar*, para el caso de *$entreno = no*, llama a la función *recortar*, proporcionándole como argumentos el "marco" de la imagen, Tx, Ty y Tolerancia. Si *$entreno = si*, antes de hacer lo anteriormente mencionado, activa a la función *procesar* que se encuentra en el módulo *nusuario* para que llame a la función *recortar* hasta cumplir con las condiciones explicadas en la sección 3.1.6. En ambos casos al finalizar la función mandará como valores de retorno el número de letras y líneas encontradas en la imagen y crea un archivo llamado *letras.txt* en donde se muestra el resultado de la segmentación incluyendo mensaje que indican donde se encontró un espacio o cambio de línea. En dicho archivo las letras segmentadas son representadas en la forma de 0 (blanco) y 1(negro) como se muestra en la figura 3.2.2.3.

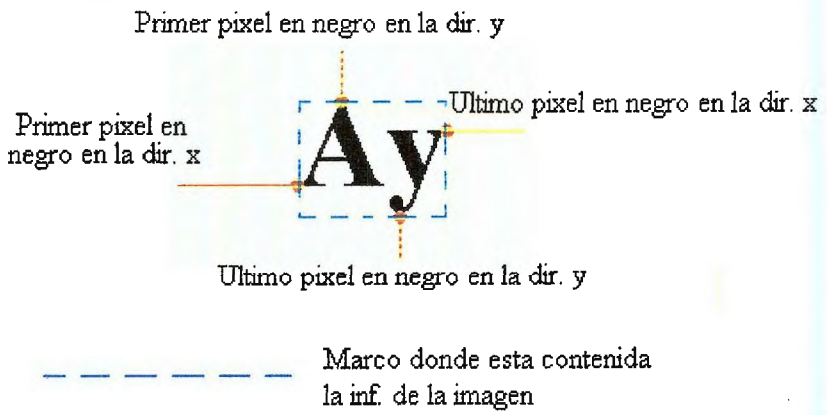


Figura 3.2.2.2. Ejemplo de un "marco" usando las posiciones de los pixeles en la función *CreaMatriz*.

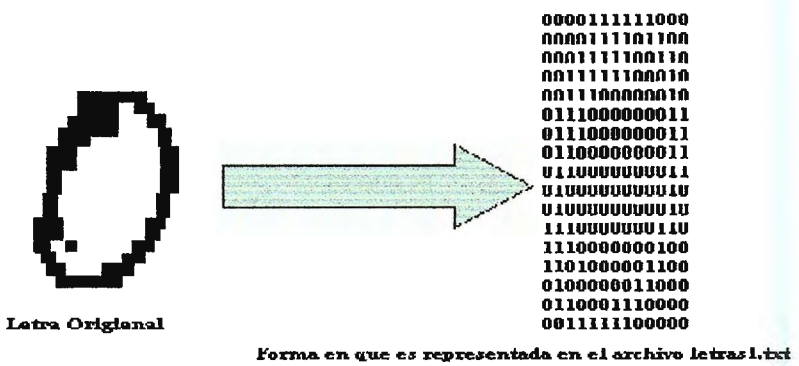


Figura 3.2.2.3. Representación de una imagen segmentada en el archivo *letras.txt*, hecho por el módulo *segmentar*.

Los caracteres que no son visibles, 'espacio' y 'cambio de línea', son identificados en el archivo *letras.txt* con las etiquetas 'espacio' e 'Intro' respectivamente. Cada vez que se ha segmentado una letra su representación lleva al final la etiqueta 'letra', con lo cual se sabe que la siguiente línea de información pertenece a otro caracter. Estas convenciones son utilizadas a lo largo de todo el proceso, así en los archivo *letras2.txt* y *destino.txt* aparecen de la misma manera, pero las etiquetas 'espacio' e 'Intro' son ignoradas durante el proceso de entrenamiento, por lo que no afectan a la red neuronal.

3.2.2.1 Rutinas ocupadas en el segmentador

A continuación se explica en detalle el funcionamiento de cada rutina ocupada por el módulo *segmentar*:

- **MostrarI:** Recibe como argumento la dirección y nombre de la imagen que se quiere trabajar. Muestra una ventana donde aparece la imagen y un botón de comando, *procesar,* al ser presionado este botón examina el valor de la variable *$entreno* de la IGU, para saber si llamará a la función *procesar* del módulo *segmentar* o de *nusuario*.

- **procesar**[13]**:** Toma los valores de *$tx, $ty* y *$tol* de la IGU. Llama a la función *CreaMatriz*, si dicha función regresa el valor de "verdadero" la imagen esta en blanco o no posee pixeles en color negro y por tanto presenta el mensaje de "¡Imagen en blanco!" en la caja de texto secundaria de la IGU, dando por terminado el módulo *segmentar*, si por el contrario el valor regresado es "falso" indica que hay información en la imagen pasada por el usuario y por tanto prosigue con la segmentación, llama a la función *recortar* de la siguiente manera: *($letras,$lineas) = &recortar($Primerx, $Primery, $Ultimox, $Ultimoy, $tx, $ty, $tol);*. La función recortar regresa el número de letras y de líneas, se le debe proporcionar las posiciones del "marco" y del tamaño de la ventana de muestreo (Tx y Ty) además de la tolerancia a ocupar para su funcionamiento.
- **CreaMatriz:** Se encarga de crear la matriz que contiene toda la imagen, además llama a otras 4 funciones para ubicar el "marco" de la imagen. Primero toma el alto y ancho de la imagen con lo que se crea un lazo para asignar cada valor de la imagen a una posición de *GranMatriz* que contiene todos los datos. Cada pixel de la imagen posee los 3 componentes de color (RGB) por lo que es necesario convertir los datos a 0 (pixel en blanco) y 1 (pixel en negro), solo se examina el valor de R (rojo), esto debido a que el blanco es todos los colores con sus valores máximos a la vez y negro es todos los colores a 0, por tanto al examinar solo uno de los colores, y tomando en cuenta que solo se introducirán imágenes binarizadas entonces es más que suficiente para saber si el pixel es de color negro o blanco, si el color R es 0 entonces el pixel se toma como negro y se le asigna el valor de 1 en caso contrario es blanco y se le asigna el valor de 0, el resultado es almacenado en una posición de *GranMatriz*. Las funciones que llama *CreaMatriz* para conocer los límites del "marco" de la imagen son *PrimerP_y, PrimerP_x, UltimoP_y* y *UltimoP_x*. La función *PrimerP_y* es la primera que se llama y además se usa para saber si la imagen se encuentra en blanco, ya que si no encuentra ningún pixel en negro entonces regresa el valor de "imagen en blanco", con lo que se termina la función *CreaMatriz*. Si en caso contrario la imagen tiene información regresa el valor de la posición en *y* del primer pixel, las restantes 3 funciones también regresan las posiciones y *CreaMatriz* regresa el valor de "falso" indicando que si hay información en la imagen.
- **PrimerP_y:** Encuentra la posición del primer pixel en negro en un escaneo de arriba hacia abajo, de izquierda a derecha, regresa un valor escalar que equivale a la posición en la dirección *y* o la hilera "imagen en blanco".

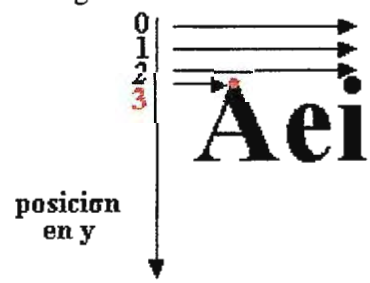


Figura 3.2.2.1.1. Ejemplo de muestreo, para este caso se regresaría el valor 3, pues es en esa posición donde se encontró el primer pixel en negro en la dirección y.

[13] La función *procesar* del módulo *nusuario* se explica en la sección 3.1.6.1.

- **PrimerP_x:** Encuentra la posición en *x* del primer pixel en negro, funciona de manera similar a la función anterior, con la diferencia de que su escaneo es de izquierda a derecha, de arriba hacia abajo. Al encontrar un pixel regresa el valor de su posición. Esta función no tiene necesidad de revisar si la imagen esta en blanco puesto que la función anterior ya verificó esa posibilidad.

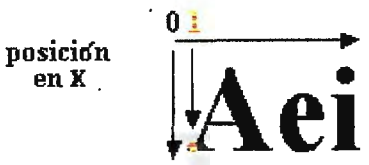


Figura 3.2.2.1.2. Ejemplo para la función *PrimerP_x*, regresaría el valor de 1.

- **UltimoP_y:** Encuentra la posición del último pixel en negro en la dirección *y*. Realiza un muestreo de abajo hacia arriba de derecha a izquierda, regresa la posición del pixel en *y*.
- **UltimoP_x:** Regresa la posición en la dirección *x* del último pixel de la imagen. Ejecuta un escaneo de derecha a izquierda de abajo hacia arriba.
- **Gemelo:** Esta función es ocupada únicamente cuando se esta creando un nuevo usuario, se encarga de copiar los datos contenidos en *GranMatriz* antes de llamar a la función *recortar*, y copiarlos en otra matriz llamada *GemeloGranMatriz*, se hace porque al usar la función *recortar* el contenido de *GranMatriz* es borrado, por tanto para no estar creando una y otra vez la matriz, buscando el "marco" de la imagen, etc. se crea otra matriz con su mismo contenido. Los datos de la matriz *GemeloGranMatriz* posteriormente son copiados de nuevo a *GranMatriz* por medio de la función *UsarGemelo*. Es llamada por la función *procesar* del módulo *nusuario*.
- **UsarGemelo:** Copia los datos contenidos en *GemeloGranMatriz* a *GranMatriz* para poder ejecutar de nuevo la función *recortar*. Es ocupada únicamente en la creación de nuevos usuarios y es llamada por la función *procesar* del módulo *nusuario*.
- **recortar:** Es la función que se encarga de dividir la imagen proporcionada según los parámetros de tamaño de ventana y tolerancia fijados. La figura 3.2.2.1.3 muestra un diagrama de flujo de parte de esta función. En la diagrama de flujo aparece que entra en un lazo, que se repetirá hasta haber examinado toda la imagen. Busca, dentro del rango establecido por Ty, un pixel en negro en la dirección *x* desde la primera posición en *x* donde encontró el último pixel (en la primera iteración este valor sería igual a la posición dada por *PrimerP_x*) hasta el valor máximo en *x* del marco (dado por *UltimoP_x*), sino encuentra ningún pixel en negro por un rango dado por Tx entonces asume que hay un espacio y sigue examinando, al llegar a la última posición en *x* que puede examinar en esa línea aumenta el valor de la posición en *y* que se esta evaluando, además aumenta el contador de líneas y equivale a encontrar el fin de una línea o un cambio de línea. Si llega hasta el final del marco (dado por *UltimoP_y*) asume que ya no hay más información se sale del lazo principal y termina *recortar*.
  Con los valores obtenidos se fijan las variables *$menorx* y *$menory*, las cuales contienen la primera posición donde se encuentra un pixel para poder ser pasado por la ventana de segmentación, el proceso de la ventana se muestra en las figuras 3.2.2.1.4 y 3.2.2.1.5, dicha ventana tiene un tamaño dado por Tx y Ty, estableciendo con ello el tamaño máximo de caracter que puede segmentar, si el caracter es más grande que esos

parámetros resultará partido hasta el tamaño de la ventana y se tomará como si fuese dos caracteres en lugar de uno solo.

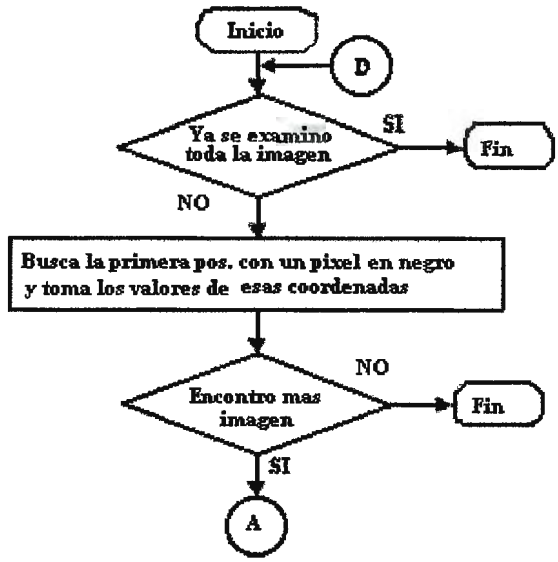


Figura 3.2.2.1.3 Inicio de la función *recortar*, entra en un lazo que termina al revisar toda la imagen. Busca un pixel en negro adentro del "marco" para tomar sus coordenadas para establecer la posición desde donde comenzará a tomar los pixeles para formar un caracter. El diagrama continúa en la figura 3.2.2.1.4.

En la figura 3.2.2.1.4 se muestra uno de los dos muestreos que se realiza en el interior del segmento dado por los valores de Tx y Ty, los cuales indican el tamaño de la ventana de muestreo, en dicha ventana se encontrará un conjunto de pixeles que aparecen conectados entre sí formando un caracter. Cada pixel que se coloca en esa ventana es obtenido por cualquiera de los dos muestreos que posee esta rutina: de arriba hacia abajo de izq. a der. o de arriba hacia abajo de der. a izq. La razón por la que se realizan estos dos muestreos es porque si solo se realizará uno de ellos podría pasar por alto algún pixel que este conectado pero que, debido a la forma en que se verifica la conectividad, es ignorado. Por tanto si el pixel aparece como conectado en alguno de los dos muestreos se coloca en la ventana de muestreo, además de colocarse en la ventana se borra el pixel de la matriz *GranMatriz* y aumenta la cuenta de pixeles encontrados. Esta cuenta de pixeles se usa para saber si se esta obteniendo información útil o simplemente algún residuo o basura, como lo podría ser algún punto, muy diminuto para considerarse un caracter, que aparece en la imagen y debe ser ignorado, sino apareciese esta restricción se podría tomar como un caracter. La cantidad de pixeles que se considera información útil viene dado por el valor de las tolerancias sumadas y después multiplicadas por 0.8, si la cuenta de pixeles es menor que ese valor la información que aparece en la ventana es ignorada y no aumenta la cuenta de letras encontradas ni es puesta en el archivo de salida del segmentador ('letras.txt'). Esta verificación se realiza después de haberse efectuado los dos muestreos para ese segmento de la imagen.

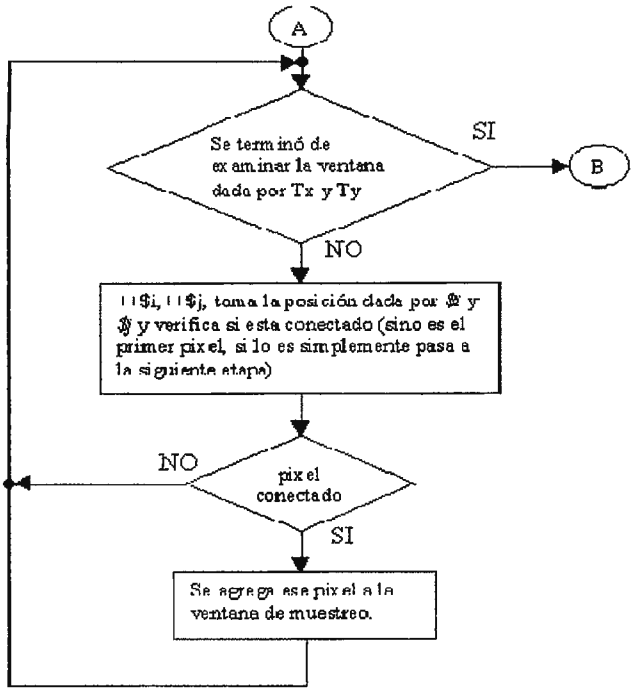


Figura 3.2.2.1.4 Diagrama de flujo para el muestreo de arriba hacia abajo de der. a izq.

Un ejemplo de lo que sucede al muestrearse la imagen únicamente de una manera puede verse en la figura 3.2.2.1.6. Se borran los pixeles ya calificados como conectados de *GranMatriz* para evitar que interfieran en los procesos para encontrar otros pixeles que todavía no se han tomado en cuenta, de lo contrario al examinar la imagen en la primera parte, donde se establece la posición del primer pixel para ser muestreado, quedaría en un lazo infinito y no pasaría del mismo sector. No es perjudicial debido a que solo se borran los pixeles que aparecen como conectados, podría decirse que ya han sido "usados", y por ende no contienen más información útil. Cada vez que se encuentra un pixel conectado se varia el tamaño que posee la ventana, pues esta varía según la cantidad de pixeles encontrados, por lo que se tienen algunas variables que se encargan de almacenar las posiciones máximas y mínimas de la ventana donde se posee información, pues solo es adentro de estas posiciones donde se encuentra la información relevante y que será impresa en el archivo "letras.txt". En la figura 3.2.2.1.6 puede apreciarse como se va realizando el muestreo de arriba hacia abajo de izq. a der., se puede notar como el pixel que se encuentra en la parte inferior, en la primera columna y última línea de la imagen es ignorado aunque esta conectado al resto de la imagen y este adentro de la ventana de muestreo. Esto debido a que se busca conectividad con respecto a los pixeles de la ventana y no de *GranMatriz*, pues de esta manera se probó que tenía una mejor segmentación ya que al verificar en *GranMatriz* se tiene el problema que puede tomar pixeles que no están todavía comprobados que pertenecen al caracter para comprobar la conectividad. Por lo que podría decir que un pixel esta conectado aún cuando estuviese comprobando conectividad con pixeles que no tienen nada que ver con el caracter, estropeando con ello la segmentación.

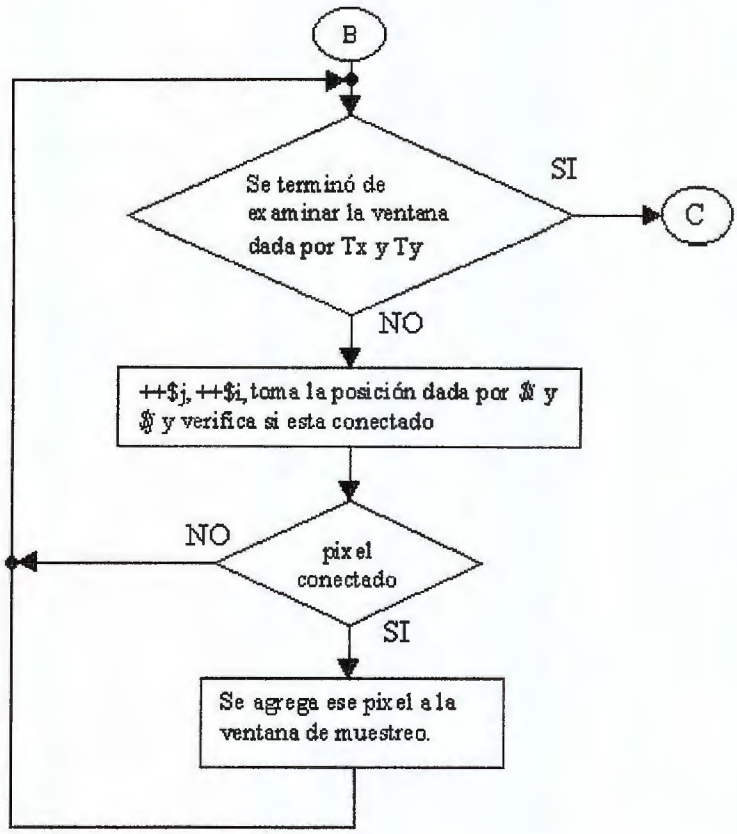


Figura 3.2.2.1.5 Diagrama de flujo para el muestreo de arriba hacia abajo de izq. a der., para este caso ya no se examina si es el primer pixel porque no puede pasar por la etapa anterior sin haber encontrado al menos un pixel.

Figura 3.2.2.1.6 Se muestra como se realiza un muestreo de abajo hacia arriba de izq. a der. Las líneas con número par (2 y 4) poseen el contenido de la ventana de muestreo, las impares (1 y 3) tienen la información de *GranMatriz*. Al final se puede ver que no se obtienen todos los pixeles de imagen aunque estén conectados. Se asume un valor de tolerancia de 1 y un Tx y Ty lo suficientemente grande para contener todo el caracter. Dos pixeles están conectados si aparecen adentro del cuadro rojo en las líneas pares.

Al finalizar el muestreo se puede observar, de la figura 3.2.2.1.6 que no toda la imagen es tomada, se puede evitar parte del error pasando varias veces el muestreo en la misma

dirección pero se comprobó, a través de varias pruebas, que la mejor solución era pasar otro muestreo en la dirección contraria, esto debido a que el primer muestreo toma los trazos de izq. a der. pero aún quedan algunos trazos de der. a izq. que son ignorados, los cuales se toman con el segundo muestreo, como lo muestra la figura 3.2.2.1.7. Todos los pixeles que habían sido ignorados y que pertenecían al caracter son tomados, completando, con ello, la segmentación del caracter de manera satisfactoria.

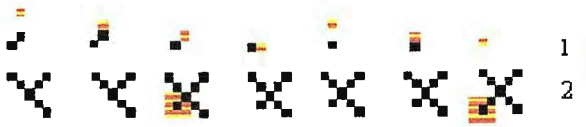


Figura 3.2.2.1.7 Al pasar el muestreo de der. a izq. se toman los datos que fueron ignorados en el anterior muestreo, completándose el caracter en la ventana de muestreo.

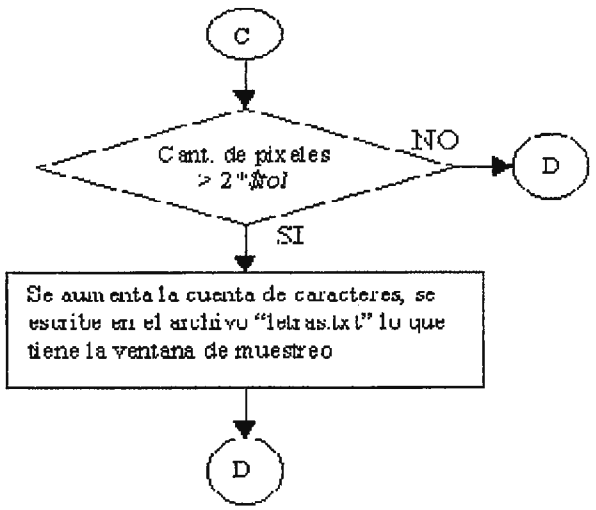


Figura 3.2.2.1.8 Diagrama de flujo del proceso siguiente al muestreo.

Nombre de la función:	Cometido
MostrarI	Muestra la imagen, que será procesada, en una ventana.
procesar	Llama a otras funciones para realizar la segmentación.
CreaMatriz	Crea una matriz con los límites y la información de la imagen.
PrimerP_y	Encuentra la posición en *y* del primer pixel en negro.
PrimerP_x	Encuentra la posición en x del primer pixel en negro.
UltimoP_y	Encuentra la posición en *y* del último pixel en negro.
UltimoP_x	Encuentra la posición en *x* del último pixel en negro.
Gemelo	Crea una matriz con la información que tiene *GranMatriz*
UsarGemelo	Pasa los datos de *GemeloGranMatriz* a *GranMatriz*.
recortar	Divide la imagen en segmentos donde se presume hay un caracter, la información es grabada en un archivo en forma de ceros y unos.

Tabla 3.2.2.1.1 Funciones del módulo segmentar

Para la función *recortar* se habla de buscar conectividad entre los pixeles, lo cual consiste en verificar si el pixel que se esta evaluando en ese momento se encuentra en la cercanía de alguno de los pixeles en negro que se encuentran ya en la ventana de muestreo. La distancia máxima a la que puede estar separado el pixel evaluado de algún pixel en negro esta relacionado por la tolerancia. Los diagramas de flujo para el proceso de verificación de conectividad se muestran en las figuras 3.2.2.1.9 y 3.2.2.1.10. El proceso de verificación de conectividad se realiza para ambos muestreos (de izq. – der., arriba – abajo y de der. – izq., arriba – abajo), siendo independiente el de un muestreo con el otro, aunque ambos procesos son idénticos.

En la figura 3.2.2.1.9 se puede observar que se necesita obtener la posición relativa del pixel que se esta verificando, esto debido a que ese pixel ocupa una determinada posición en *GranMatriz*, pero la ventana de muestreo tiene un tamaño máximo dado por Tx y Ty, por tanto el valor de su posición debe encontrarse entre esos valores, por lo que se recurre a ocupar la posición que ocuparía ese pixel si solo se tomase el caracter al que se supone pertenece como si ocupase una matriz del mismo tamaño que la ventana de muestreo. Para una mejor ilustración ver la figura 3.2.2.1.11.

En la figura 3.2.2.1.11 se muestran las posiciones en *GranMatriz* que forma una imagen, la letra T mayúscula, y como al ser pasada a la ventana de muestreo los pixeles ocupan una posición distinta a la que ocupaban en *GranMatriz*, pero siempre guardando la relación que existe entre ellos para formar la misma letra T en la ventana de muestreo. El tamaño de la ventana de muestreo viene dado por Tx y Ty, y para este caso se asume que son lo suficientemente grandes como para contener toda la imagen.

Con este arreglo se asegura que se mantendrá el contenido de la imagen, pero que no será necesaria otra matriz del mismo tamaño que *GranMatriz* para realizar cada segmentación.

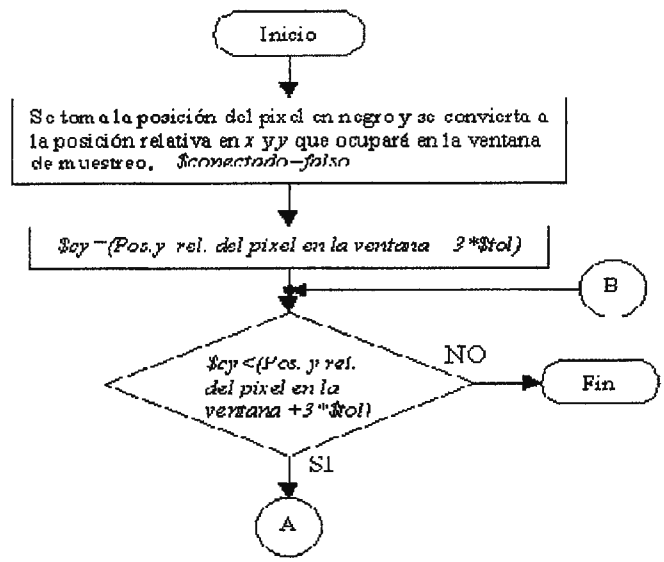


Figura 3.2.2.1.9. Primera parte del diagrama de flujo para encontrar la conectividad entre dos pixeles.

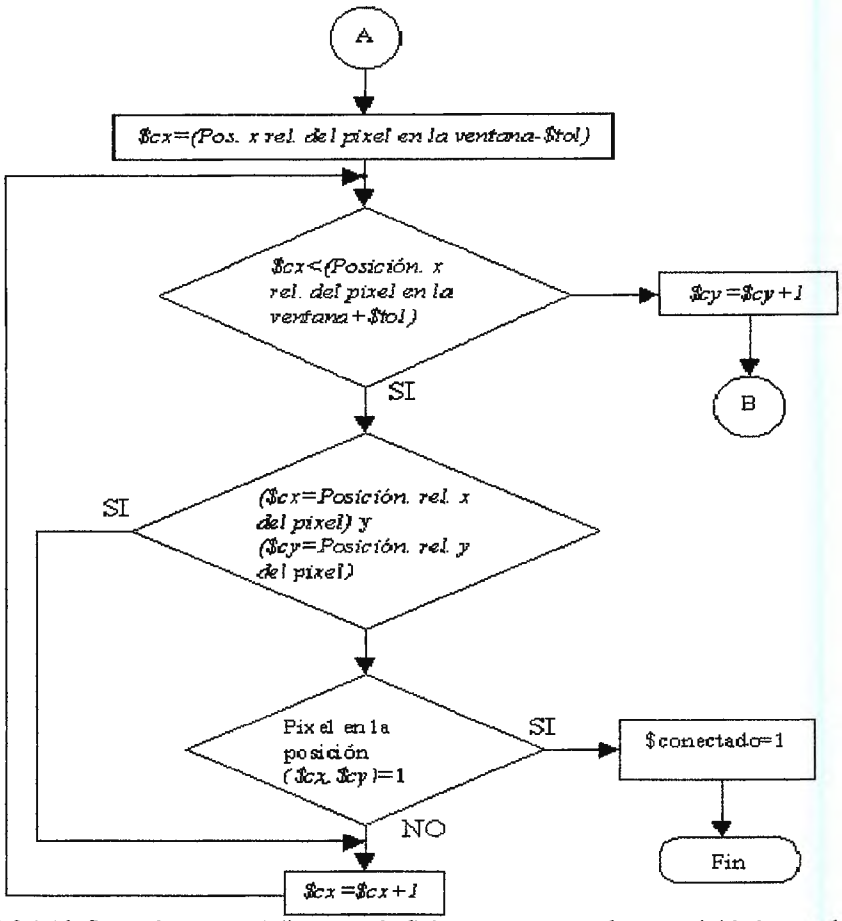


Figura 3.2.2.1.10. Segunda parte del diagrama de flujo para encontrar la conectividad entre dos pixeles.

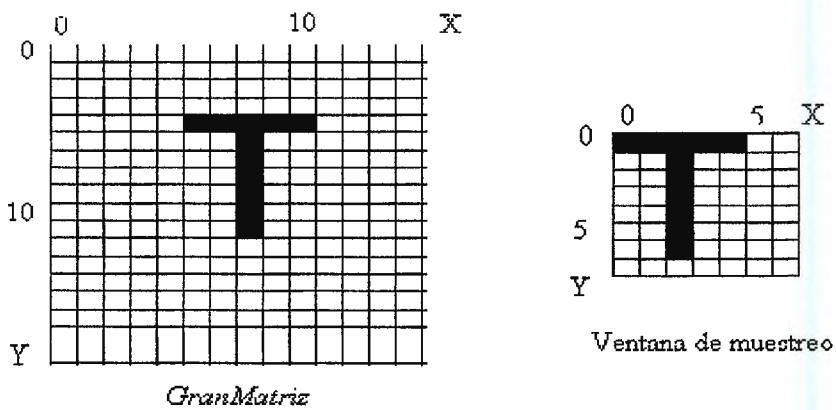


Figura 3.2.2.1.11. Se muestra las posiciones relativas que ocuparía los pixeles de una imagen en la ventana de muestreo con respecto a las que originalmente tenían en *GramMatriz*. El pixel que en *GranMatriz* ocupaba la posición 5,5 en la ventana de muestreo ocupa la posición 0,0.

3.3 RED NEURONAL

El tipo de red neuronal que se ha escogido para el reconocimiento es la red ART2[14] (Teoría de Resonancia Adaptativa), la cual trabaja con valores analógicos. Se necesita que sea así porque se piensa agregar una etapa de procesamiento de los datos procedentes del segmentador, para mejorar la clasificación por parte de la red neuronal.

Este tipo de red neuronal presenta además las ventajas de elasticidad, requiere poco entrenamiento y solo puede existir una sola salida activa, pues es una red autocompetitiva, donde sus salidas compiten entre ellas para establecer a que clasificación pertenece el patrón de datos que se encuentra en su entrada.

El número de entradas y salidas no podrá ser modificado por el usuario, cuando el programa este terminado. Por motivos de demostración y experimentación por el momento todos los parámetros de la red pueden ser modificados (por ejemplo: número de entradas y salidas, parámetros de tolerancia).

3.3.1 Introducción teórica de las redes ART[15]

La red ART se presenta como una solución al problema de *estabilidad y plasticidad del aprendizaje* que planteó *S. Grossberg*:

- Cómo una red podría aprender nuevos patrones (plasticidad del aprendizaje).
- Cómo una red podría retener los patrones previamente aprendidos (estabilidad del aprendizaje).

Con la ART (Teoría de Resonancia Adaptativa) se tiene un sistema competitivo (red con aprendizaje competitivo) con el cual si se presenta cierta información de entrada sólo una de las neuronas de salida de la red (o una por cierto grupo de neuronas) se activa alcanzando su valor de respuesta máximo después de competir con las otras. Esta neurona recibe el nombre de *vencedora* (*winner – take – all unit*).

Se pretende categorizar los datos que se introducen en la red. Las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida, la neurona vencedora. La teoría de resonancia adaptativa se basa en la idea de hacer *resonar* la información de entrada con los representantes o prototipos de las categorías que reconoce la red. Si entra en resonancia con alguno, es suficientemente similar, la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo almacenado representante de la categoría para que se incorpore algunas de las características del dato presentado.

---

[14] Para una ampliación del concepto ART2 consultar las referencias 1 y 5.
[15] Tomado de las referencias 1 y 5

Al principio se crearon dos tipos de ART: la red ART 1 (para datos digitales) y la red ART2 (para datos analógicos), actualmente hay otros tipos de ART pero no serán tomadas en cuenta en este trabajo.

La ART1 presenta 2 capas entre las que se establecen conexiones hacia delante y hacia atrás. Las conexiones entre las neuronas de salida tienen pesos tales que se inhiben unas a otras para que exista competencia (ver Figura 3.3.1.1).

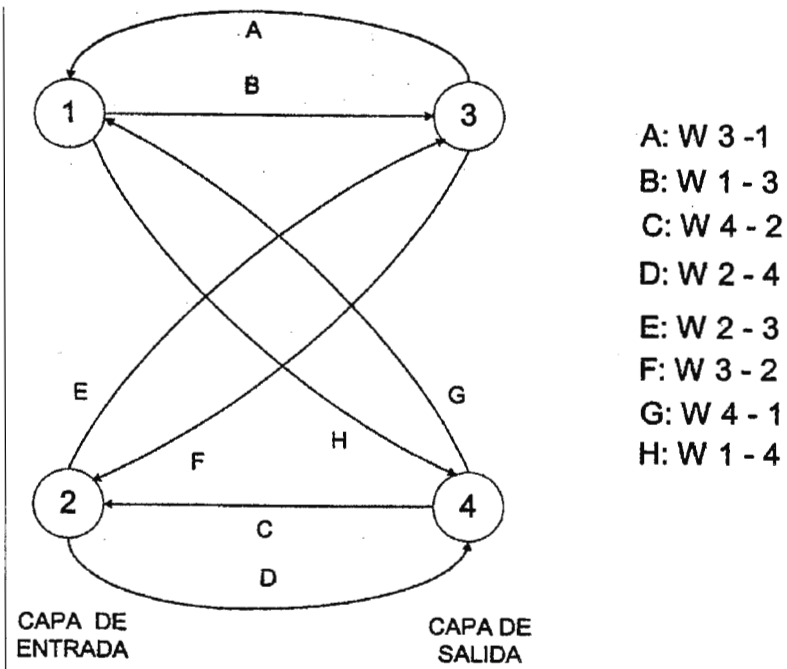


Figura 3.3.1.1 Topología de una red ART simple, se muestran su capa de entrada, de salida y los pesos ascendentes y descendentes entre las neuronas. Tomado de las clases de Sistemas de Control Automático 2.

La red ART2 tiene la misma estructura de una red ART, pero más compleja ya que dentro de su capa de entrada y salida se encuentran diferentes subcapas de tratamiento. Estas subcapas se encargan de reducir el vector de entrada a su vector unitario en algunos casos. Como la red ART2 procesa señales análogas se deben reducir estos vectores a un vector unitario para poder clasificarlo en una categoría. El proceso de clasificación se lleva a cabo mediante el vector resultante de entrada contra los pesos.
El proceso de la ART2 es el siguiente:
La red consiste de dos partes F1 y F2. M es el número de neuronas de F1 y N el número de neuronas de salida. Las siguientes condiciones se deben cumplir para seleccionar los otros parámetros. Las constantes a, b, c, d, e, ρ y θ son valores de ajuste para la red. Donde ρ es el parámetro de vigilancia y θ el mínimo valor de entrada análoga detectable.

$a, b > 0$ (Ec. 3.3.1.1)
$0 \leq d \leq 1$ (Ec 3.3.1.2)
$cd / (1-d) \leq 1$ (Ec 3.3.1.3)
$0 \leq \theta \leq 1$ (Ec 3.3.1.4)
$0 \leq \rho \leq 1$ (Ec 3.3.1.5)

$$e << 1 \qquad \text{(Ec. 3.3.1.6)}$$

Todos los pesos descendentes tienen valores iniciales de cero.

$$Z_{ij}(0) = 0 \qquad \text{(Ec. 3.3.1.7)}$$

Todos los pesos ascendentes iniciales tienen el siguiente valor:

$$Z_{ij}(0) \leq 1 / ((1-d)\sqrt{M}) \qquad \text{(Ec. 3.3.1.8)}$$

El procesamiento de la señal una vez se introduce el dato es:

1) Se ponen en cero los vectores de las capas y subcapas del sistema, se inicializa un contador con uno.
2) Se aplica el vector análogo de entrada I_i a la capa W de F1.

$$W_i = I_i + aU_i \qquad \text{(Ec. 3.3.1.9)}$$

3) W se propaga a la subcapa X.

$$X_i = W_i / \left(e + \|W\|\right) \qquad \text{(Ec. 3.3.1.10)}$$

El vector se reduce a su vector unitario X sí e=0.

4) X se propaga a la capa V.

$$V_i = f(X_i) + bf(Q_i) \qquad \text{(Ec. 3.3.1.11)}$$

En donde, $f(X) = \begin{cases} 0 & 0 \leq X \leq \theta \\ X & X > \theta \end{cases}$

$f(Q) = \begin{cases} 0 & 0 \leq Q \leq \theta \\ Q & Q > \theta \end{cases}$

5) Se propaga a la capa U.

$$U_i = V_i / \left(e + \|V\|\right) \qquad \text{(Ec. 3.3.1.12)}$$

6) Se propaga a la subcapa P.

$$P_i = U_i + \delta Z_{ij} \qquad \text{(Ec. 3.3.1.13)}$$

En el proceso de aprendizaje δZ_{ij} es cero, porque los pesos son cero. En los demás casos, cuando ya se han definido pesos para los diferentes nodos de salida, se multiplica "d" por la columna de pesos correspondientes a la salida seleccionada, las demás columnas son iguales a cero.

Donde, $\delta Z_{ij} = G(T_j) \bullet Z_{ij}$.

7) P se propaga a la subcapa Q.

$$Q_i = P_i / \left(e + \|P\|\right) \qquad \text{(Ec. 3.3.1.14)}$$

8) Se repiten los pasos del 2 al 7 hasta que se estabilicen los datos. Generalmente dos veces es suficiente.

9) Se calcula la salida de la capa R.

$$R_i = (U_i + cP_i)/\left(e + \|U\| + c\|P\|\right) \qquad \text{(Ec. 3.3.1.15)}$$

10) Se determina si hay restauración si, $\rho / \left(e + \|R\|\right) > 1$, por lo tanto se restaura F2 y se marcan todos los posibles nodos activos de F2 como no aptos para la competición regresándose al paso 2. Al no cumplirse la ecuación y el contador es igual a 1. Se incrementa el contador y se sigue con el paso 11. Si no se cumple la ecuación y el contador es igual a 2 se salta al paso 14 porque hay resonancia (el vector de entrada es lo suficientemente parecido al vector de referencia de una categoría).
Se entiende como restauración, al proceso en el cual se inicializan todos lo valores de capas y subcapas. Eliminando el vector actual de entrada, esperando por uno nuevo para comenzar en el paso 1. Porque el vector de entrada actual no puede ser catalogado en ninguna categoría existente.

11) Se propaga P hasta F2.

$$T_j = \sum P_i Z_{ij} \quad \text{Para todas las filas} \qquad \text{(Ec. 3.3.1.16)}$$

12) Se selecciona el nodo ganador basándose en lo siguiente no tomando en cuenta nodos marcados para no competir.

$G(T_j) = d$ $\quad T_j$ es máximo para todo T mientras sea valido el nodo
0 $\quad$ En caso contrario

13) Se repiten los pasos 6 al 10.

14) Se modifican los pesos ascendentes de la unidad ganadora F2.

$$Z_{Ji} = U_i / (1-d) \qquad \text{(Ec. 3.3.1.17)}$$

15) Se modifican los pesos descendentes de la unidad ganadora.

$$Z_{iJ} = U_i / (1-d) \qquad \text{(Ec. 3.3.1.18)}$$

16) Se elimina el vector de entrada y se ponen a cero los valores de capas y subcapas. Inicializando en el paso 1.

Se puede observar el diagrama de flujo del proceso en la figura 3.3.1.2.

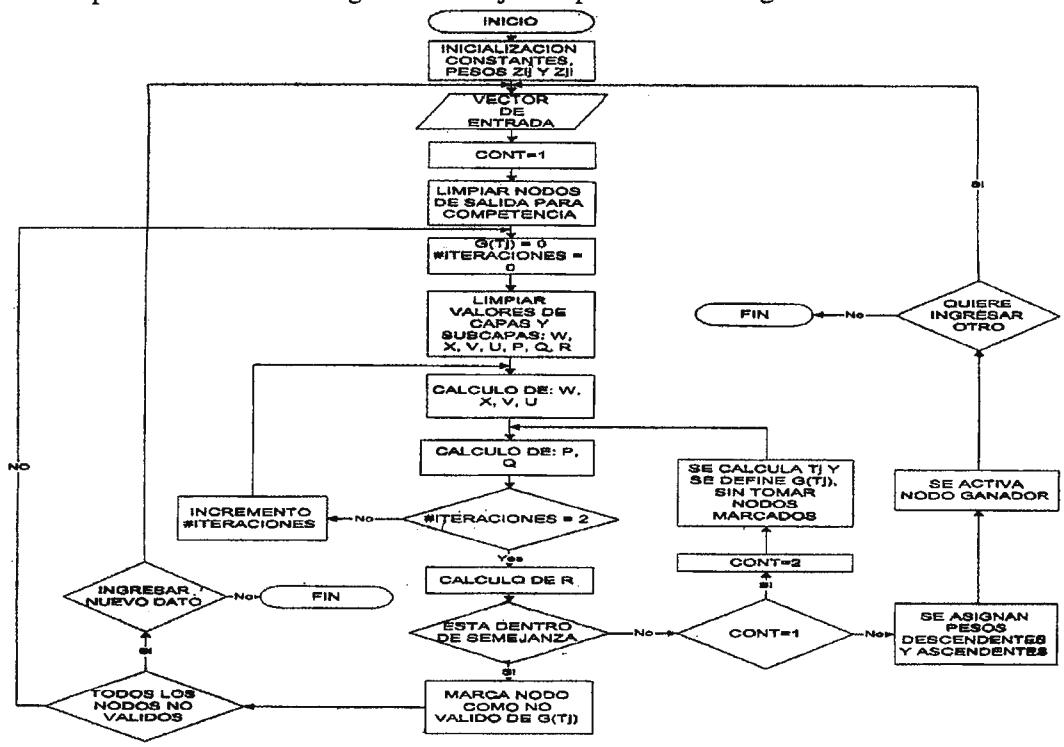


Figura 3.3.1.2 Diagrama de flujo del proceso de la ART2, tomado de las clases de Sistemas de Control Automático 2, redes ART.

3.3.1.1 Ejemplo del proceso de la red ART2

Las constantes tomaran los siguientes valores: a=10; b=10; c=0.1; d=0.9; e=0; θ=0.2; ρ=0.9. El vector de entrada es I_i= (0.2, 0.7, 0.1, 0.5, 0.4). La capa F2 o de salida tiene N=6 y F1 o de entrada tiene M=5.

En primer lugar se inicializan los pesos ascendentes y descendentes, de la siguientes forma. Pesos descendentes igual a cero.

$$Z_{ij} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Pesos ascendentes, $Z_{ij} = \frac{0.5}{(1-d)\sqrt{M}} = \frac{0.5}{(1-0.9)\sqrt{5}} = 2.236$

$$Z_{ij} = \begin{matrix} 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \end{matrix}$$

El proceso es el siguiente a través de las capas y subcapas:

1) Se propaga el vector de entrada a través de las capas y subcapas, debido a que en este momento no hay realimentación todos los vectores existentes en las capas y subcapas son cero. Y el contador es igual a 1.
2) Se propaga I a W.

$W_i = I_i + aU_i = (0.2, 0.7, 0.1, 0.5, 0.4) + 10(0, 0, 0, 0, 0) = (0.2, 0.7, 0.1, 0.5, 0.4)$

3) W se propaga a la subcapa X.

$$X_i = W_i / (e + \|W\|)$$
$$\|W\| = \sqrt{(0.2^2 + 0.7^2 + 0.1^2 + 0.5^2 + 0.4^2)}$$
$X_i = (0.2, 0.7, 0.1, 0.5, 0.4) / (0+0.9746)$
$X_i = (0.205, 0.718, 0.103, 0.513, 0.410)$

4) X se propaga a la capa V.

$V_i = f(X_i) + bf(Q_i)$

En donde, $f(X) = 0 \quad 0 \le X \le \theta$
$X \quad X > \theta$
$f(Q) = 0 \quad 0 \le Q \le \theta$
$Q \quad Q > \theta$

$V_i = f(0.205, 0.718, 0.103, 0.513, 0.410) + 10f(0, 0, 0, 0, 0)$
$V_i = (0.205, 0.718, 0.000, 0.513, 0.410)$

5) Se propaga a la subcapa U.

$U_i = V_i / (e + \|V\|) = (0.205, 0.718, 0.103, 0.513, 0.410) / (0+0.9943)$
$U_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

6) Se propaga hasta la subcapa P.

$P_i = U_i + \delta Z_{ij}$ como $Z_{ij} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$, ver paso 6 del proceso ART2.

$P_i = U_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

7) P se propaga a la subcapa Q.

$$Q_i = P_i / (e + \|P\|) = (0.206, 0.722, 0.000, 0.516, 0.412) / (0+0.9998)$$
$Q_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

8) Se comienza la segunda iteración para estabilizar los datos desde el paso 2 al 7.

a) Se propaga I a W.

$W_i = I_i + aU_i = (0.2, 0.7, 0.1, 0.5, 0.4) + 10(0.206, 0.722, 0.000, 0.516, 0.412)$
$W_i = (2.26, 7.92, 0.1, 5.66, 4.52)$

b) W se propaga a la subcapa X.

$$X_i = W_i / (e + \|W\|) = (2.26, 7.92, 0.1, 5.66, 4.52) / (0+10.9685)$$
$X_i = (0.206, 0.722, 0.009, 0.516, 0.412)$

c) X se propaga a la capa V.

$V_i = f(X_i) + bf(Q_i)$

En donde, $f(X) = \begin{cases} 0 & 0 \le X \le \theta \\ X & X > \theta \end{cases}$

$f(Q) = \begin{cases} 0 & 0 \le Q \le \theta \\ Q & Q > \theta \end{cases}$

$V_i = f(0.206, 0.722, 0.009, 0.516, 0.412) + 10f(0.206, 0.722, 0.000, 0.516, 0.412)$
$V_i = (2.266, 7.942, 0.000, 5.676, 4.532)$

d) Se propaga a la subcapa U.

$$U_i = V_i / (e + \|V\|) = (2.266, 7.942, 0.000, 5.676, 4.532) / (0+10.9984)$$
$U_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

e) Se propaga hasta la subcapa P.

$P_i = U_i + \delta Z_{ij}$ como $Z_{ij} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$, ver paso 6 del proceso ART2.

$P_i = U_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

f) P se propaga a la subcapa Q.

$Q_i = P_i/(e+\|P\|) = (0.206, 0.722, 0.000, 0.516, 0.412) / (0+0.9998)$
$Q_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

Dos iteraciones son suficientes para estabilizar los datos. Se sigue adelante.

9) Se calcula la salida de la capa R.

$R_i = (U_i + cP_i)/(e+\|U\|+c\|P\|) = ((0.206,\ 0.722,\ 0.000,\ 0.516,\ 0.412) + 0.1(0.206, 0.722, 0.000, 0.516, 0.412)) / (0 + 0.9998 + 0.1(0.9998))$
$R_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

10) Se determina si hay restauración (ver paso 10 de proceso ART2) sí $\rho/(e+\|R\|) > 1$, por lo tanto se restaura F2 y se marcan todos los posibles nodos activos de F2 como no aptos para la competición regresándose al paso 2. Al no cumplirse la ecuación y el contador es igual a 1. Se incrementa el contador y se sigue con el paso 11. Si no cumple la ecuación y el contador es igual a 2 se salta al paso 14 porque hay resonancia.

$\rho/(e+\|R\|) = 0.9/(0+1) = 0.9$ no es mayor que 1, por lo tanto se incrementa contador y se sigue adelante.

11) Se propaga P hasta F2.

$T_i = \sum P_i Z_{ii} = (0.206, 0.722, 0.000, 0.516, 0.412)$

$\begin{matrix} 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \end{matrix}$

2.236 2.236 2.236 2.236 2.236

Se multiplica cada valor de P_i con su respectivo valor de la primera fila de Z_{ji}, sumando todos los productos se genera el primer valor de T_j. De igual manera pero cambiando de fila Z_{ji} (pasar a la segunda, tercera...) para generar todos los valores de T_j.

$T_j = (4.152, 4.152, 4.152, 4.152, 4.152, 4.152,)$

12) Se selecciona el nodo ganador basándose en lo siguiente no tomando en cuenta nodos marcados para no competir.

$G(T_j) = d$ T_j es máximo para todo T mientras sea valido el nodo
0 En caso contrario.

Como T_j tiene todos los componentes iguales y no se a marcado a ninguno para no competir, se selecciona el primero de izquierda a derecha como ganador. Quedando la función G de la siguiente manera.

$G(T_j) = (0.9, 0, 0, 0, 0, 0)$

13) Se repiten los pasos 6 a 10.

a) Se propaga hasta la subcapa P.

$P_i = U_i + \delta Z_{ij}$
$P_i = U_i$ pero para visualizar bien el desarrollo plantearemos como se calcula δZ_{ij}

$$\delta Z_{ji} = G(T_j)Zji = (0.9, 0, 0, 0, 0, 0) \bullet \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} = (0, 0, 0, 0, 0, 0)$$

Se desarrolla el producto punto de dos matrices, multiplicando respectivamente los valores de la matriz 1 con la primera columna de la matriz 2. Sumando luego los productos, para obtener el primer componente de δZ_{ij}. Se repite el proceso con la siguiente columna de la matriz de Z_{ij} para obtener el segundo componente de δZ_{ij} y así sucesivamente. Para poder calcular P_i.

$P_i = (0.206, 0.722, 0.000, 0.516, 0.412) + (0, 0, 0, 0, 0, 0)$
$P_i = (0.206, 0.722, 0.000, 0.516, 0.412)$

b) P se propaga a la subcapa Q.

$$Q_i = P_i / \left(e + \|P\|\right) = (0.206, 0.722, 0.000, 0.516, 0.412) / (0+0.9998)$$

Q_i = (0.206, 0.722, 0.000, 0.516, 0.412)

c) Como el contador tiene 2 se sigue adelante, no es necesario hacer mas iteraciones.

d) Se calcula la salida de la capa R.

$R_i = (U_i + cP_i)/(e + \|U\| + c\|P\|)$ = ((0.206, 0.722, 0.000, 0.516, 0.412) + 0.1(0.206, 0.722, 0.000, 0.516, 0.412)) / (0 + 0.9998 + 0.1(0.9998))

R_i = (0.206, 0.722, 0.000, 0.516, 0.412)

e) Se determina si hay restauración (ver paso 10 proceso ART2) sí $\rho/(e + \|R\|) > 1$, por lo tanto se restaura F2 y se marcan todos los posibles nodos activos de F2 como no son aptos para la competición regresándose al paso 2. Al no cumplirse la ecuación y el contador es igual a 1. Se incrementa el contador y se sigue con el paso 11. Si no se cumple la ecuación y el contador es igual a 2 se salta al paso 14 porque hay resonancia.

$\rho/(e + \|R\|) = 0.9/(0+1) = 0.9$ no es mayor que 1, el contador es igual a 2 se salta al paso 14.

14) Se modifican los pesos ascendentes de la unidad ganadora de F2.

$Z_{ji} = U_i/(1-d)$ = (0.206, 0.722, 0.000, 0.516, 0.412) / (1 – 0.9)

Z_{ji} = (2.06, 7.22, 0.00, 5.16, 4.12)

Z_{ij} =

2.06	7.22	0.00	5.12	4.12
2.236	2.236	2.236	2.236	2.236
2.236	2.236	2.236	2.236	2.236
2.236	2.236	2.236	2.236	2.236
2.236	2.236	2.236	2.236	2.236
2.236	2.236	2.236	2.236	2.236

15) Se modifican los pesos descendentes de la unidad ganadora.

$Z_{ji} = U_i/(1-d)$ = (0.206, 0.722, 0.000, 0.516, 0.412) / (1 – 0.9)

Z_{ji} = (2.06, 7.22, 0.00, 5.16, 4.12)

Z_{ij} =

2.06	0	0	0	0	0
7.22	0	0	0	0	0
0.00	0	0	0	0	0
5.16	0	0	0	0	0
4.12	0	0	0	0	0

Se elimina el vector de entrada y se ponen a cero los valores de capas y subcapas. Inicializando en el paso 1. Para esperar un nuevo vector de entrada.

3.3.2 Descripción del módulo ART2.

Se ha creado un módulo para la red ART2, llamado *art2.pm*, que sigue los pasos descritos en la sección 3.3.1, con la adición de que se pueden cargar pesos desde un archivo y guardar los pesos que se obtienen como resultado del proceso.

Los datos de entrada para el módulo deben ser mayores o iguales que cero, una vez ingresados los datos el módulo se encarga de realizar un lazo hasta encontrar una respuesta o que todas las neuronas hallan quedado fuera de competencia.

Cada paso descrito en la sección anterior se ha convertido en una función, teniendo funciones extra para realizar la división de un vector entre un escalar y la magnitud de un vector, las cuales son muy usadas a lo largo del proceso.

Las cantidades de salidas, así como los parámetros *a, b, c, d, e,* θ, y ρ son definidos en el archivo *configuraart2.ini*, el cual debe encontrarse ubicado en el directorio de trabajo del módulo. El número de entradas es necesario pasarlo como argumento al momento de inicializar la red. Esto último debido a que la cantidad de entradas depende del resultado de multiplicar *$tx*$ty* en la fase de entrenamiento, este resultado es almacenado en el archivo de parámetros del usuario para su uso durante el funcionamiento normal del programa (fuera del entrenamiento).

3.3.3 Programación del módulo ART2.

El módulo de ART 2 que se ha implementado posee el diagrama de flujo mostrado en la figura 3.3.3.1.

Para iniciar se llama a la función *art2_inicio*, pasándole el número de entradas como argumento, la cual se encarga de configurar la red (número de entradas, salidas, pesos iniciales), cuando la red ya esta configurada se puede llamar a la función para leer los pesos desde un archivo.

Con la red y sus pesos listos se pasan los datos a ser evaluados por la red, dichos datos estarán separados por comas o en forma de un arreglo (una matriz que posee una serie e valores que son independientes entre sí y pueden ser accesados individualmente), y pueden tener cualquier valor que sea mayor o igual a cero. La cantidad de datos debe ser igual al número de entradas de la red ART. Después de pasar los datos se inicia el proceso en sí de la red, que trata de entrar en resonancia con los datos de entrada y de dar una respuesta. La respuesta que puede dar la función *art2_procesa* es un escalar que puede ser un número (la neurona ganadora) o el mensaje de que se puede clasificar la entrada.

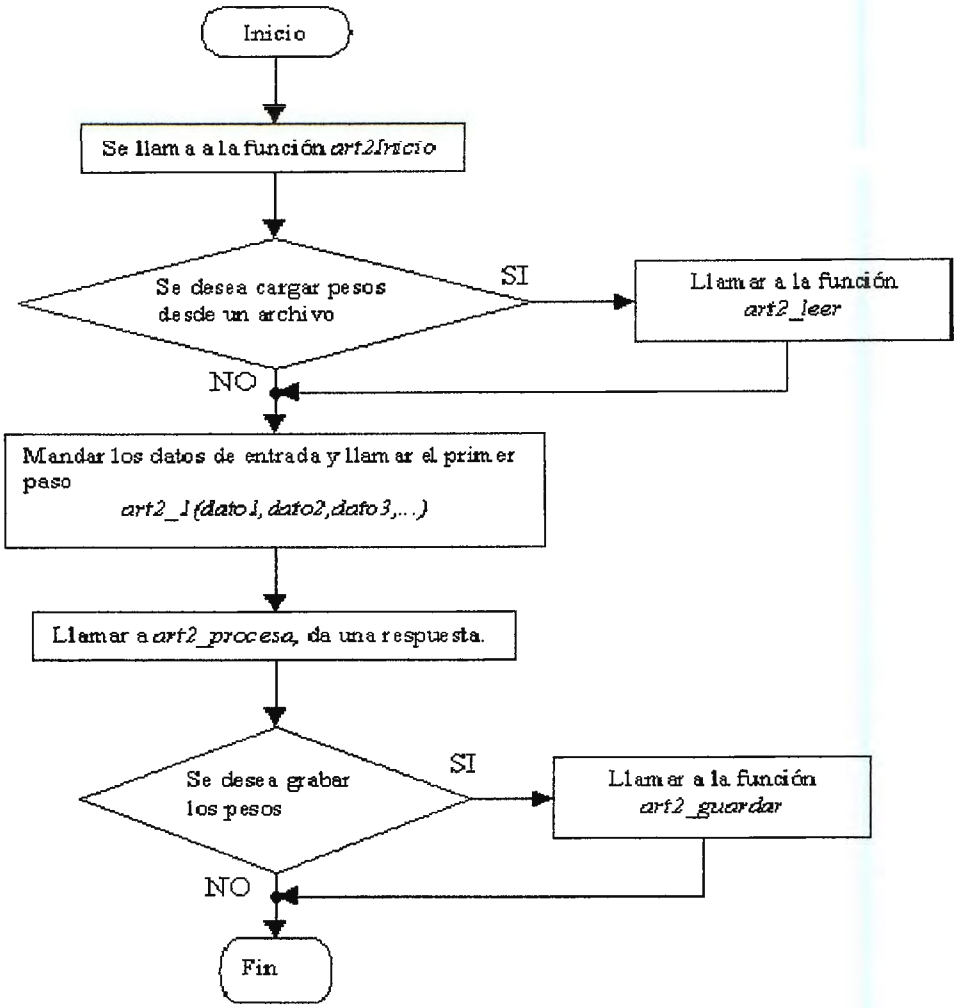


Figura 3.3.3.1 Diagrama de flujo del módulo de ART2.

Para el módulo de la ART 2 cada paso descrito en la sección 3.3.1 se ha convertido en una función, por tanto lo que hace la función *art2_procesa* es llamarlas en la secuencia apropiada, evaluar si ya se tiene una respuesta o aún existen neuronas para seguir compitiendo, cuando ya no hay neuronas para competir entonces regresa el mensaje de que la entrada no se ha podido clasificar.

La función *art2_procesa* no llama a los pesos grabados en algún archivo ni tampoco se encarga de grabarlos. Por tanto si se desea realizar alguna de esas acciones debe llamarse a la función apropiada del módulo ART 2.

Al final se puede llamar a la función para grabar los pesos si se desea guardar el resultado del proceso.

El llamar los pesos desde un archivo o leerlos es opcional, al llamar a los módulos para leer o guardar los pesos se debe proporcionar el nombre del archivo del que se leerá o donde se guardarán los datos.

3.3.3.1 Rutinas ocupadas en el módulo de ART 2

A continuación se describen todas las funciones que posee este módulo, argumentos que necesitan, salida (si es que devuelven algo), y, de ser necesario, diagramas de flujo para mostrar los procesos, las funciones desde la *art2_1* hasta la *art2_15* son equivalentes a los pasos descritos en la sección 3.3.1 para el proceso de la ART 2:

- **art2Inicio:** Se encarga de configurar la red ART 2, lee los parámetros de ajuste de la red así como la cantidad de entradas y salidas que posee. Toda esa información la toma del archivo *configuraart2.ini* que debe encontrarse en el directorio de trabajo, excepto por la cantidad de entradas que debe pasarse como argumento al momento de invocar esta función, esto debido a que la cantidad de datos de entrada es variable pues depende de cómo queden los parámetros *$tx* y *$ty* en el proceso de creación de un usuario, se necesita que la red tenga tantas entradas como pixeles pueda contener la multiplicación de esos dos parámetros, es decir debe tener *$tx*$ty* entradas. Además pone los pesos iniciales de la red, como fue establecido en la sección 3.3.1 en la inicialización de la red. Se inicializa el arreglo *@GanAnt*, el cual tiene significado únicamente durante el proceso de creación de un usuario, este arreglo se encarga de controlar cuales han sido las salidas que se han activado, esto con el objeto que durante el entrenamiento se asegure que se activa una salida distinta para cada caracter, evitando de esta manera que queden salidas sin usar. Durante el funcionamiento normal del programa este arreglo es ignorado y pueden activarse cualquier salida el número de veces que sea necesario.

- **art2_1:** Se llama a esta rutina para pasarle los datos de entrada a ser procesados por la red (*art2_1(dato1,dato2,dato2...);*), inicializa algunas variables como el contador de iteraciones (*$contador=1*), los nodos que se encuentran fuera de competencia (*@NodosFuera=()*), la neurona ganadora (*$NG=-10*).
- **art2_2:** Se inicializan los vectores que son ocupados por el módulo (*@W, @X, @V, @U, @P, @Q, @R, @T, @G, @sigmaZiJ*), pasa el vector de entrada I a W, tal como se describe en el paso 2.
- **art2_3:** Llama a la función *mag1D* y le pasa como argumento el vector *@W*, devolviendo la magnitud del vector pasado como argumento, que sería ||W||. Después llama a la función *VentE* que divide un vector entre un escalar, en este caso sería *@W/($e+||W||)*, donde *$e* es el valor del parámetro *e*. El resultado se pasa al vector *@X*.
- **art2_4:** Se pasa el vector *@X* a la capa *V*, esto se hace con un condicional metido en un lazo, para evaluar tanto a *$X* como a *$Q*, de la manera que se muestra en la figura 3.3.3.1.1. Donde se ve que si el valor de *$X* o de *$Q* es menor que *$theta* el resultado de la variable evaluada será cero, en caso contrario conservan su valor y se suman para formar el valor de *$V*, pero antes el valor de *$Q* es multiplicado por *$b*.

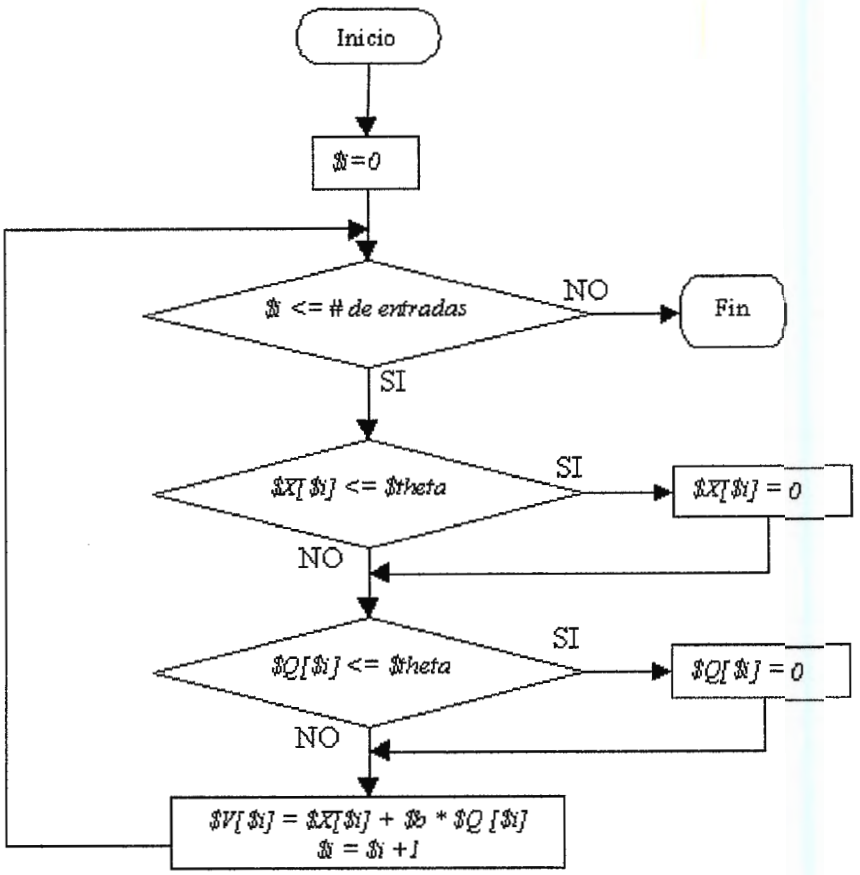


Figura 3.3.3.1.1 Diagrama de flujo para la rutina *art2_4*.

- **art2_5:** Se pasa el vector *@V* hacia *@U*, se usan las funciones *mag1D* y *VentE* para realizar el paso 5 en el proceso de la ART 2. Funciona de manera similar a la función *art2_3*.
- **art2_6:** Se propaga *@U* hacia *@P*, como se describe en el paso 6 si existe una neurona ganadora, identificada como cualquier valor distinto de −10 para la variable *$NG*, se multiplica cada valor de G por la primera fila de *$Zij*, se suman los productos y se obtiene el primer valor de *$sigmaZiJ* y se continúa con la siguiente fila de Zij. El valor de −10 en *$NG* únicamente indica que todavía no se ha escogido un ganador. El diagrama de flujo se muestra en la figura 3.3.3.1.2.
- **art2_7:** *@P* se propaga hacia *@Q*, se llama a las funciones *mag1D* y *VentE* para poder realizar la operación descrita en el paso 7 del proceso de la ART 2. Para la función *mag1D* se pasa como argumento a *@P* para poder obtener su magnitud, al ser obtenida se le suma el valor de *$e* y se pasa como argumento para la función *VentE*, también se manda el vector *@P* a dicha función para poder realizar la división de un vector entre un escalar.
- **art2_8:** Se llaman de nuevo a las funciones: *art2_2, art2_3, art2_4, art2_5, art2_6* y *art2_7*. Esto se hace para estabilizar los datos de los arreglos usados a lo largo del proceso.

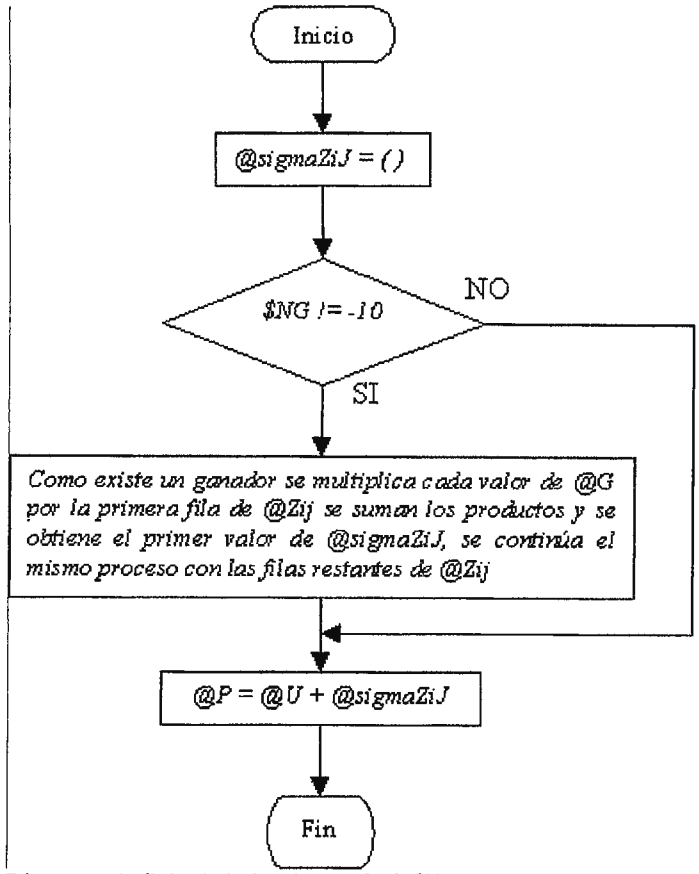


Figura 3.3.3.1.2 Diagrama de flujo de la función *art2_6*, *$NG* es igual a –10 cuando no se ha encontrado algún ganador.

- **art2_9:** Se calcula *@R*. Primero se obtiene el valor *$magP* al llamar a la función *magnitud1D* y pasarle el arreglo *@P* como argumento, después el valor de *$magU* que es igual al valor obtenido de la función *magnitud1D* pasándole el arreglo *@U* como argumento. Se calcula el valor de *$Esc* sumando los valores *$e* + *$magP* + *$magU*, para después llamar a la función *VentE* con *$Esc* y *@U* como argumentos, con lo que se consigue el valor de *@divU*, de la misma manera se obtiene el valor de *@divP* pero usando *@P* en lugar de *@U* en el argumento de la función *VentE*. Finalmente *@R* es igual a sumar *@divU* más *$c* * *@divP*.
- **art2_10:** Esta rutina regresa un valor cuando es llamada, dicho valor puede ser "Restauración", que indica la desactivación de la salida y la repetición del proceso, "Todos Fuera", cuando todos los nodos han quedado fuera de competencia o un valor entero, que indica la neurona de salida que se ha activado. Determina si hay restauración (sucede cuando la salida de la capa R no es lo suficientemente parecida a la entrada), si la hay, desactiva el nodo que había sido activado en la salida y no compite más en el proceso, se pasa a la función *art_Procesar* el valor "Restauración", para que repita el proceso de tratar de hacer que la red entre en resonancia, pero tomando en cuenta que el nodo ha sido desactivado, se repite el proceso desde la función *art2_2*. Si hay resonancia realiza el proceso descrito en figura 3.3.3.1.3. Donde se puede observar que lo primero que se revisa es el valor de *$contador*, esto debido a que si vale 1 indica

que es la primera vuelta y es necesario realizar una iteración más para comprobar si la red entra en resonancia (es lo suficientemente parecido el valor almacenado en la red con la entrada que se esta evaluando) y llama a la función *art2_11* para propagar el valor de *@P* hacia F2 para activar una salida, después llama a *art2_12* para seleccionar el nodo ganador, esta última función regresa un valor ("Todos Fuera" o el número del nodo ganador) que es evaluado por la función *art2_10* para terminar el proceso o llamar a la función *art2_13;* si la variable *$contador* vale 2 indica que ya se comprobó que la salida activada es la correcta, por tanto llama a las funciones *art2_14* y *art2_15* que se encargan de variar los pesos asociados a la neurona ganadora.

- **art2_11:** Se propaga *@P* hacia la capa de salida, esto se hace multiplicando la primera fila de *@Zij* con los valores de *@P* se suman los productos obtenidos y el resultado es el primer valor de *@T*, se realiza el mismo proceso multiplicando *@P* por cada fila de *@Zij*.
- **art2_12:** Se selecciona el nodo ganador o regresa el mensaje de que todos los nodos están fuera de competencia. Encuentra la primera neurona que puede competir (no ha sido marcada como desactivada por el programa), sino encuentra alguna neurona que pueda competir termina el proceso y regresa el mensaje 'Todos Fuera". Cuando se esta creando un nuevo usuario además se toma en cuenta el contenido de *@GanAnt* en la determinación de un ganador, si la neurona ya ha sido escogida como ganadora en alguna de las ocasiones anteriores su posición en el arreglo *@GantAnt* tendrá el valor de "1", por tanto no puede ser escogida otra vez y se reanuda la búsqueda de una salida ganadora, esto sucede únicamente si el programa esta creando un usuario, de lo contrario ignora el contenido de *@GanAnt*. Para saber si se esta creando un nuevo usuario o esta en funcionamiento normal esta función verifica la variable *$entreno* de la *IGU*. Si encuentra alguna neurona que pueda competir compara el valor *@T* que posee en la posición de la neurona escogida con el valor de la siguiente neurona que puede competir para guardar la posición de la neurona cuyo valor en *@T* sea el mayor, una vez obtenida la posición de la neurona ganadora se establece el valor de *@G* el cual es cero para cualquier posición que no sea el de la neurona escogida, para la posición de la neurona escogida el valor de *@G* es igual a *$d*. Regresa la posición de la neurona escogida.
- **art2_13:** Es llamada por la rutina *art2_10* y se encarga de realizar una iteración más para saber si la red entra en resonancia. Puede regresar cualquiera de los siguientes tres valores: 'Todos Fuera", 'Restauración" o la posición de la neurona ganadora. Primero llama a la función *art2_6* para propagar de nuevo *@U* hacia *@P*, después llama a *art2_7* para propagar *@P* hacia *@Q*, posteriormente llama a la función *art2_8* para estabilizar los datos de los vectores, se procede con la función *art2_9* para calcular *@R* y al final se obtiene el valor al llamar a la función *art2_10*, que deberá regresar el valor final del proceso, debido a que se iría al caso en que el *$contador* vale 2 y el valor encontrado es el correcto, o se recibió el mensaje de 'Restauración" o "Todos Fuera", los cuales al ser regresados a la rutina *art2_10* que llamó a la rutina *art2_13* terminará el proceso o marcará el nodo como no válido para que se repita el proceso sin que ese nodo pueda competir.
- **art2_14:** Si se llama a esta función es porque se ha encontrado un nodo ganador y entro en resonancia, al cumplirse estas condiciones se procede a modificar los pesos ascendentes de la unidad ganadora (Z_{ji}). El valor que poseerán los pesos se realiza

como esta descrito en el paso 14 de la sección 3.3.1. Se realiza un lazo para modificar los pesos que se encuentran en la columna *$NG* de la matriz *@Zji*.

- **art2_15:** Ahora se modifican los pesos descendentes de la unidad ganadora por medio de esta función, el valor de los nuevos pesos se describe en el paso 15 de la sección 3.3.1. Con un lazo se modifican los pesos de la fila *$NG* de la matriz *@Zji*.
- **art2_guardar:** Graba los pesos en un archivo cuyo nombre es especificado como un argumento de esta función. Primero abre o crea el archivo con el nombre especificado, después realiza dos procesos, grabar los pesos ascendentes en orden separados cada columna por un espacio en blanco y cada fila por un retorno de carro y una alimentación de línea, después realiza otro retorno de carro y otra alimentación de línea para escribir los pesos descendentes con las mismas separaciones que los pesos ascendentes. El archivo creado esta en formato texto.
- **art2_leer:** Lee los pesos almacenados en un archivo que tiene el nombre especificado como argumento al realizar la llamada de esta función. Esta función puede ser llamada después de haber inicializado la red y, aunque la red ya posee pesos iniciales, poder de esta manera cargar los pesos que se han obtenido al entrenar a la red. También realiza dos procesos para leer los pesos de la red, primero lee los pesos ascendentes (Zji) tomando en cuenta las separaciones descritas en la función *art2_guardar*, después lee los pesos descendentes (Z_{ij}), todos los pesos son asignados a la red en lugar de los pesos que tuviera en ese momento. Los pesos anteriores al llamado de esta función se pierden.
- **art2_procesa:** Después de haber inicializado la red con la función *art2Inicio* y pasarle los datos a clasificar con la función *art2_1*, se llama a ésta función para que controle el resto del proceso de la red ART 2. Se encarga de llamar a las otras funciones para ejecutar los siguientes pasos y de brindar una respuesta ("No clasificable" o el número de nodo donde se clasificaron los datos), además se encarga de finalizar el proceso de la ART, entrar en un lazo llamando continuamente a las funciones en caso de que la entrada no halla sido clasificada en la primera iteración y aún queden nodos para competir. El diagrama de flujo de la función *art2_procesa* se muestra en la figura 3.3.3.1.4. Se inicializa la variable *$Fin* con el valor "restauración", el valor de esa variable se modifica con lo que regresa la función *art2_10*, ese valor puede ser "restauración" si no se clasifico la entrada pero aún quedan nodos para competir, "Todos Fuera" si no se pudo clasificar y ya no quedan nodos para competir y un número si se pudo clasificar indicando en que salida se clasifico el dato. A partir de ese resultado se toma la decisión de realizar otra iteración, terminar el proceso regresando "no clasificable" o regresar el valor del nodo donde fue clasificada la entrada.

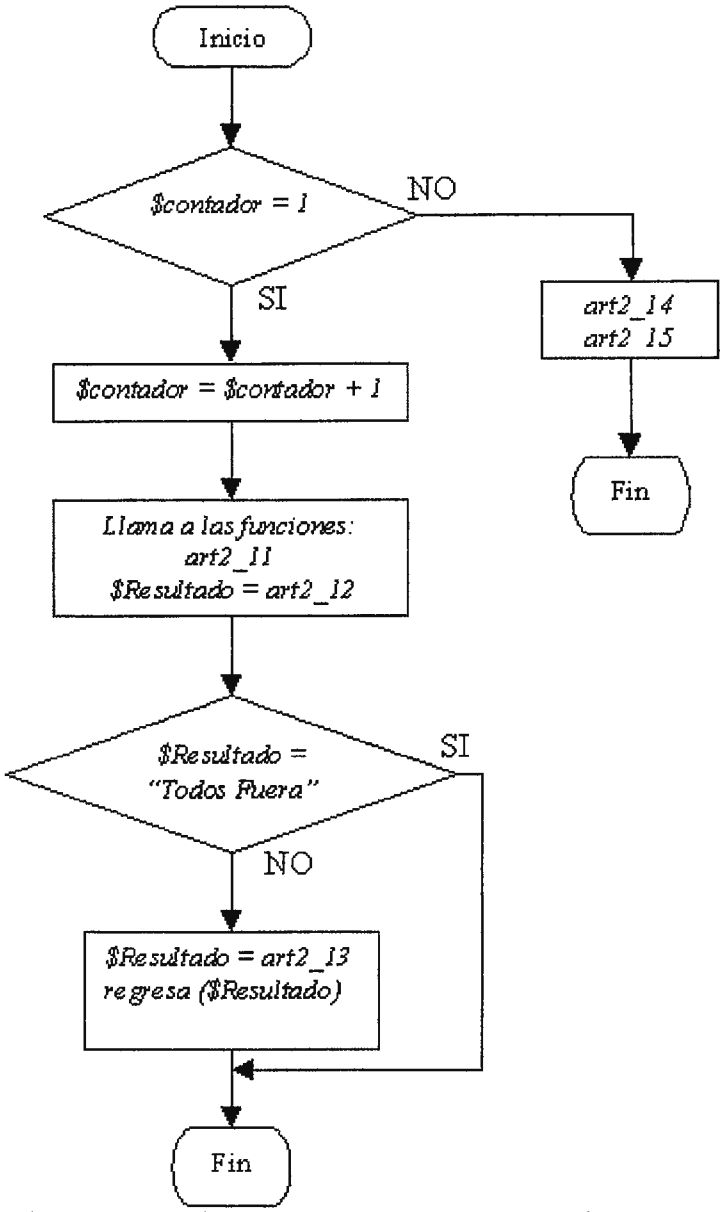


Figura 3.3.3.1.3 Diagrama de flujo del proceso de la función *art2_10* cuando hay resonancia.

- **mag1D:** Se ocupa para encontrar la raíz cuadrada de la suma de los cuadrados de los datos pasados como argumento. Esta función es bastante utilizada a lo largo del proceso de la ART 2. Los argumentos que se le pasan a esta función son elevados al cuadrado para posteriormente ser sumados, al resultado de esa suma se le aplica la función raíz cuadrada, el valor obtenido de esa operación es regresada por esta función, dicho valor es un escalar, la entrada es un vector de tamaño arbitrario.
- **VentE:** Es otra función ocupada en el proceso de la ART 2. Debido a que el programa Perl no puede operar vectores o arreglos directamente es necesario usar esta función para poder dividir un vector entre un escalar. Esta función recibe dos argumentos, el vector que se quiere dividir, y el escalar que servirá como divisor. Un detalle importante

es que los argumentos que se pasan en una función en Perl se convierten en un solo vector, con lo que los datos que se pasan con esta función se convierten en un solo vector, por lo que se debe cuidar que el último dato del vector sea el escalar pues la función extrae el último valor del argumento de entrada asumiendo que ese es el divisor. Con el dato extraído del argumento lo demás entra en un lazo por medio del cual se divide cada uno de los valores del vector entre el escalar obtenido previamente. Al final regresa un vector.

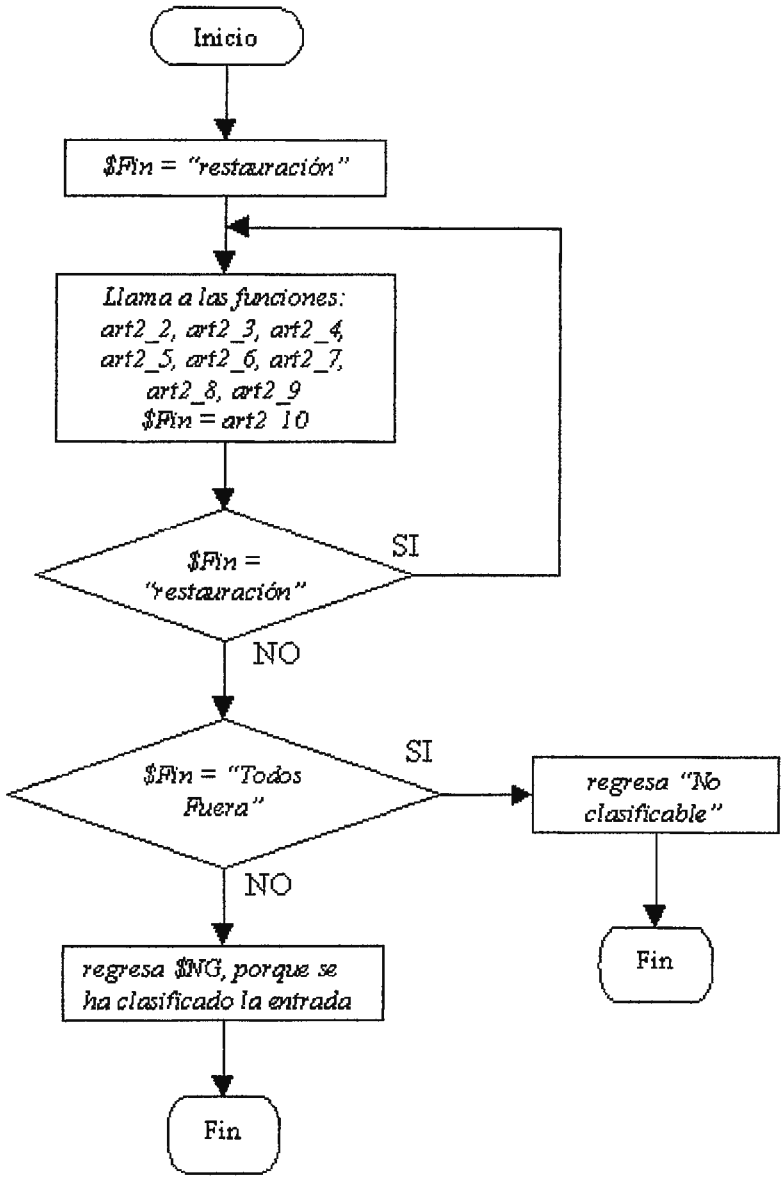


Figura 3.3.3.1.4. Diagrama de flujo de la función *art2_procesa*, que se encarga de manejar el proceso de la ART 2.

Nombre de la función	Cometido
art2Inicio	Configura los parámetros iniciales de la red ART, los cuales lee de un archivo.
art2_1	Se pasan los datos de entrada, inicializa algunas variables.
art2_2	Inicializa los vectores ocupados por el módulo. Pasa el vector de entrada I a W.
art2_3	Pasa W a X.
art2_4	Pasa X a V.
art2_5	Pasa V hacia U.
art2_6	Se propaga U hacia P.
art2_7	Propaga P hacia Q.
art2_8	Llama a las funciones *art2_2, art2_3, art2_4, art2_5, art2_6* y *art2_7.*
art2_9	Se calcula R.
art2_10	Regresa un valor ("Restauración", "Todos Fuera" o un número) y llama a las funciones *art2_11, art2_12, art2_13, art2_14* y *art2_15.*
art2_11	Se propaga P a la capa de salida.
art2_12	Se selecciona el nodo ganador o regresa el mensaje de que todos los nodos están fuera de competencia.
art2_13	Realiza una iteración más para saber si la red entra en resonancia, llama a las funciones *art2_6, art2_7, art2_8, art2_9* y *art2_10*, esta última regresa un valor que a su vez es transmitido a la función que llamó a la *art2_13.*
art2_14	Modifica los pesos ascendentes de la unidad ganadora.
art2_15	Modifica los pesos descendentes de la unidad ganadora.
art2_guardar	Graba los pesos de la red en un archivo.
art2_leer	Lee los pesos almacenados en un archivo para ponérselos a la red.
art2_procesa	Esta función se llama después de haber inicializado la red para que se encargue del resto del proceso.
mag1D	Devuelve la raíz cuadrada de la suma de los cuadrados de los datos pasados como argumentos.
VentE	Divide un vector entre un escalar.

Tabla 3.3.3.1.1 Funciones que posee el módulo de la ART 2.

3.4 PREPROCESAMIENTO DE LA IMAGEN

3.4.1 Métodos de preprocesamiento.

Para el programa ROCM se probaron diversos métodos de preprocesamiento para suministrarle a la red ART datos significativos y útiles para clasificar correctamente los caracteres.

Entre los métodos probados se encuentran: Transformada Discreta de Coseno en dos dimensiones, suministrar los datos directamente a la red sin tratamiento previo, realizar sumas de filas e hileras para tomar el total de pixeles en negro por fila y columna, y otros más.

Para el caso de la Transformada Discreta de Coseno (TDC) en dos dimensiones se hicieron diversas variaciones pues los resultados obtenidos de la clasificación no eran satisfactorios para el caso de letra manuscrita, pues la letra de imprenta si obtenía altos niveles de caracteres reconocidos correctamente (entre el 70 al 100 %). Entre las variaciones implementadas se encuentra la regularización del tamaño de la imagen a trabajar, pues el resultado de la segmentación podía tener cualquier tamaño en x y y (siempre y cuando no sobrepasará los tamaños dados por *$tx* y *$ty*) volviendo más difícil el tratar con datos tan variables como es la escritura manuscrita. Se tuvo una mejora en el reconocimiento pero siempre abajo del 50%, por tanto después de muchas pruebas se optó por buscar nuevos métodos de preprocesamiento pero siempre se obtenían resultados muy bajos para la letra manuscrita pero la letra de máquina si era bien clasificada, lo que indicaba que el problema reside en las pequeñas variaciones introducidas por la escritura hecha a mano que puede parecer imperceptible para el usuario pero puede dar lugar a confusiones o mala interpretación de los datos. La gran desventaja de la TDC es que cuando se ingresaban dos imágenes muy parecidas, pero variaban en por ejemplo un pixel de los 200 que poseía en total la imagen, todos los coeficientes cambiaban en mayor o menor grado haciendo difícil el que la red pudiese realizar una identificación aceptable.

Con los otros métodos probados se tuvieron problemas similares, realizando una clasificación muy buena de los caracteres de imprenta pero un reconocimiento pobre o malo para la letra manuscrita[16].

3.4.2 Método de preprocesamiento escogido.

En base a las diversas pruebas se llegó al método que daba los mejores resultados el cual es la ampliación de los caracteres dados por el segmentador hasta llenar el tamaño dado por *$tx* y *$ty* y posteriormente la verificación del estado de los pixeles ubicados en el vecindario 8 del pixel con el que esta trabajando. La ampliación se realiza para estandarizar un poco las letras segmentadas, dado que una misma letra puede variar ligeramente de tamaño en cada ocasión, y lo segundo es para obtener los datos necesarios para ser ingresados en la red art.

El vecindario 8 se refiere a los pixeles que se encuentran alrededor del pixel que se esta evaluando, como se muestra en la figura 3.4.2.1.

La intención de tomar en cuenta los pixeles que se encuentran alrededor es recalcar que tantos pixeles en negro se encuentran conectados al pixel que se esta verificando, es como obtener el valor promedio de esa área conformada por los 9 pixeles.

[16] Para ver más acerca de las pruebas referirse al anexo A.3 Resultados de las pruebas del ROCM.

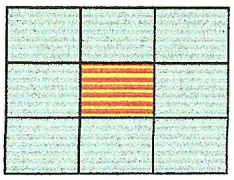


Figura 3.4.2.1. Ejemplo de un vecindario 8, el pixel en rojo es el pixel que se esta evaluando y su vecindario 8 esta conformado con los pixeles que se encuentran alrededor de él, en este caso son de color gris.

Para el ROCM se tiene que cada pixel del vecindario 8 más el pixel que esta verificando poseen un valor:

$$valor = \frac{1}{9} * color \qquad \text{Ecuación 3.4.2.1.}$$

Donde *color* toma el valor de 1 cuando el pixel que se esta evaluando de entre los 9 pixeles que se quiere saber el valor promedio, es negro y 0 cuando es blanco, por tanto si los 9 pixeles son negros se tendrá que *valor* será 1, y si solo 6 de los pixeles fuesen negros el valor obtenido sería 6/9.

3.4.3 Descripción del módulo de preprocesamiento para la ampliación de imagen (zoom)

Como se ha mencionado anteriormente los resultados del segmentador pueden variar en tamaño debido a que la letra manuscrita no es estable y una misma letra puede tener diversas formas y tamaños. Esto dificulta su clasificación y para minimizar este problema se creó el módulo de ampliación *zoom* para lograr que todos los resultados dados por el segmentador posean el mismo tamaño. En general se obliga a la imagen a "llenar" todo el espacio posible en la ventana imaginaria dada por *$tx* y *$ty*, como se muestra en la figura 3.4.3.1.

Pero hay casos que podrían crear confusión si son obligados a llenar el espacio dado por los parámetros del segmentador, por ejemplo si se ampliara el caracter "." se tendría una imagen llena de unos, pues el ampliador trataría de que el contenido de la imagen llene todo el espacio, igual problema se tendría con "-" y con la letra "l" para algunas personas, por lo tanto al final se tendría que para representar la ampliación de esos tres caracteres se tiene una imagen llena de unos, perdiéndose con ello información relevante para el reconocimiento.

Para evitar ese problema se han puesto ciertos condicionales para no aumentar en la dirección x o y o ambas una imagen. Si tiene una altura menor que 1/5 de *$ty* no se amplia en la dirección vertical, si tiene un ancho menor que 1/3 de *$tx* no se amplía en la dirección horizontal. Por tanto el punto generalmente no será ampliado en ninguna dirección y la letra "l" solo será ampliada en la dirección vertical.

Figura 3.4.3.1. Ampliación hecha por el módulo *zoom*, se han resaltado en rojo los números 1 que representan un pixel en negro en la imagen original, a la izq. se observa lo dado por el segmentador y a la derecha el resultado de la ampliación, para este caso *$tx=13*, *$ty=15*.

El módulo *zoom* procesa el archivo *letras.txt* que contiene el resultados de la segmentación, y da como salida el archivo *letras2.txt*, que contiene la misma cantidad de caracteres segmentados con la diferencia de que todos los caracteres tendrán un tamaño igual a *$tx* (ancho) y *$ty* (alto), también aparecen las etiquetas "letra", "espacio" e "Intro" tal y como aparecen en el archivo *letras.txt*.

3.4.4 Programación del módulo *zoom*

La figura 3.4.4.1 muestra el diagrama de flujo para la ampliación de caracteres. El proceso es el siguiente: en primer lugar se averiguan los valores de *$tx* y *$ty*, se abre el archivo *letras.txt*, el cual contiene el resultado de la segmentación, se verifica que lo leído no es ni la hilera "espacio" ni "Intro", si fuese alguno de ellos entonces toma ese dato y lo escribe directamente en el archivo *letras2.txt*, y se pasa a leer la siguiente información. En caso contrario lo leído es un caracter que ha sido segmentado y esta conformado por ceros y unos, esta información es asignada a la matriz *MatLetra*, que será usada en el proceso de ampliación, el resultado de ese proceso es escrito en el archivo *letras2.txt*, y después verifica si no hay más información en el archivo *letras.txt*, si no hay más información entonces termina el módulo *zoom*, de lo contrario lee otro dato y entra a verificar que dato es para saber si lo escribe sin modificarlo o si es necesario que entre en el proceso de ampliación.

En el proceso de ampliación lo primero que hace es crear una matriz llena de ceros *MatLetra2* que tiene *$tx* columnas y *$ty* filas, se averigua si es necesario ampliar en alguna dirección verificando si la cantidad de columnas en *MatLetra* es mayor que *$tx/3*, si es cierto se puede ampliar en la dirección horizontal, si la cantidad de líneas en *MatLetra* es mayor que *$ty/5* se puede ampliar en la dirección vertical. La explicación que viene a continuación asume que se puede ampliar tanto en la dirección horizontal como vertical: es necesario encontrar el factor de ampliación que tanto debe ampliarse la imagen en ambas direcciones para ocupar el espacio dado por los parámetros del segmentador *$tx* y *$ty*, lo cual se hace así:

Escala Horizontal = *$tx*/(Cantidad de columnas en *MatLetra*) Ecuación 3.4.4.1.

Escala Vertical = *$ty*/(Cantidad de líneas en *MatLetra*) Ecuación 3.4.4.2.

Con estas escalas se puede encontrar que dato en *MatLetra* corresponde a una posición en *MatLetra2*, por ejemplo si tanto la escala horizontal como vertical fuese igual a 2 entonces para las posiciones (0,0) y (0,1) en *MatLetra2* se tomaría la posición (0,0) en *MatLetra*, para las posiciones (8,8) y (8,9) en *MatLetra2* se tomaría el dato en (4,4), esto puede ser traducido a las siguientes fórmulas:

$$x_{MatLetra} = valor\ entero\ \text{de}\left(\frac{x_{MatLetra2}}{Escala\ Horizontal}\right) \quad \text{Ecuación 3.4.4.3.}$$

$$y_{MatLetra} = valor\ entero\ \text{de}\left(\frac{y_{MatLetra2}}{Escala\ \text{Vertical}}\right) \quad \text{Ecuación 3.4.4.4.}$$

Donde las posiciones en x y y encontradas con esas ecuaciones indican la posición del dato en *MatLetra* que será pasado a la posición en *MatLetra2* que se esta evaluando en ese momento.

Cuando no se realiza una ampliación en alguna dirección, por ejemplo en la dirección x, se tiene que los datos se copian directamente en esa dirección, es decir, todos los datos de la columna 1 de *MatLetra* son copiados a la columna 1de *MatLetra2* y así sucesivamente hasta llegar a la última columna en *MatLetra*, aunque pueden ser afectados en la dirección vertical pues se realizan ampliaciones independientes en ambas direcciones. Al final la matriz *MatLetra2* tendrá siempre el mismo tamaño, no importando si hubieron o no ampliaciones en alguna dirección, por lo que se obtiene un doble beneficio de este proceso, el lograr que todos los datos segmentados tengan el mismo tamaño y destacar caracteres al no ser ampliados en alguna dirección, por ejemplo regularmente la letra "l" no será ampliada en la dirección horizontal, pero si lo será en la dirección vertical, por lo que el lado izquierdo de la matriz *MatLetra2* estará conformado por columnas de ceros, aumentando su diferencia con otras letras como la "b" que aunque posee cierta semejanza al ser ampliada dicha semejanza se verá reducida pues la letra "b" si sería ampliada en la dirección horizontal hasta llenar la anchura de *MatLetra2*.

Al final del proceso todos los resultados se guardan en el archivo *letras2.txt*, que será la fuente para el módulo *vecino8*. Para el módulo de ampliación no existe mayor diferencia si se esta creando un nuevo usuario o esta funcionando normalmente, la única diferencia es que los parámetros *$tx* y *$ty* son tomados desde el módulo *nusuario* en el primer caso y desde la *IGU* cuando el programa se encuentra funcionado normalmente.

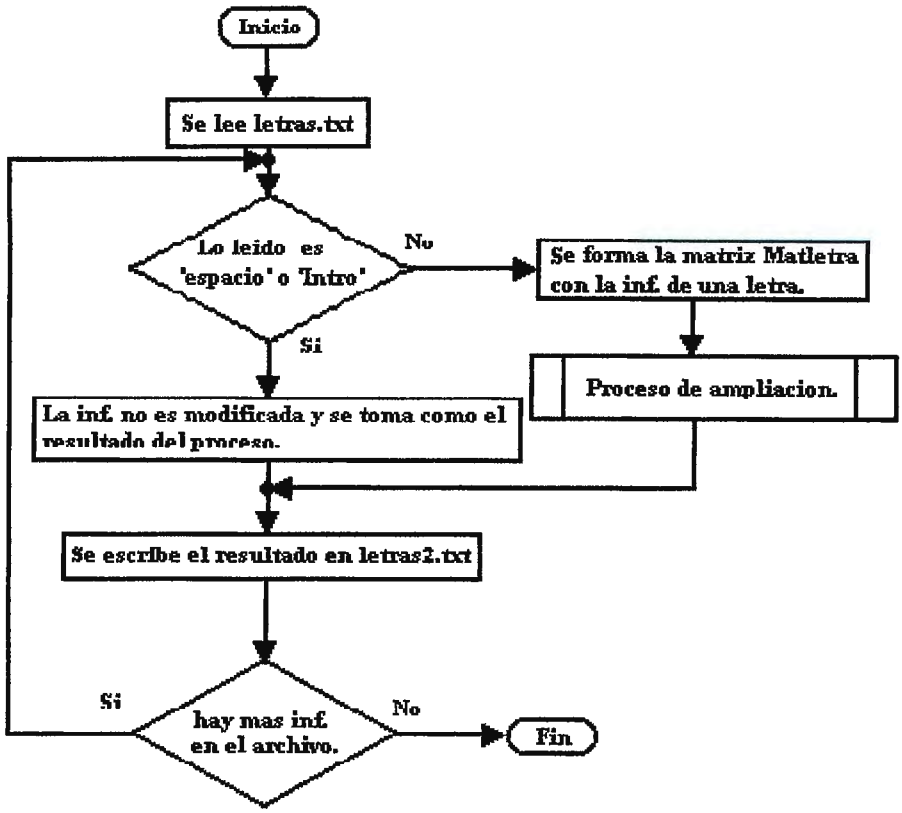


Figura 3.4.4.1. Diagrama de flujo para el módulo de ampliación *zoom*.

3.4.4.1. Rutinas ocupadas en el módulo zoom.

Las funciones ocupadas en el módulo *zoom* se describen a continuación:

- **zoom:** Es la única función del módulo y se encarga de todo el proceso de ampliación, no lleva argumentos de entrada y su salida es un archivo de texto llamado *letras2.txt*, esta función usa el archivo *letras.txt* como fuente de los datos a procesar. En primer lugar se obtienen los valores de *$tx* y *$ty*, las etiquetas "espacio" e "Intro" son copiadas directamente en el archivo de salida *letras2.txt*, el contenido de cada letra con la que se va a trabajar se almacena en la matriz *MatLetra*, se crea una matriz llena de ceros *MatLetra2* que tiene *$tx* columnas y *$ty* filas, se averigua si es necesario ampliar en alguna dirección verificando si la cantidad de columnas en *MatLetra* es mayor que *$tx/3*, si es cierto se puede ampliar en la dirección horizontal, si la cantidad de líneas en *MatLetra* es mayor que *$ty/5* se puede ampliar en la dirección vertical. El proceso de ampliación ocurre como se ha descrito en los párrafos anteriores. Esta función es invocada por el módulo *nusuario* y por *identificador*, después de que se ha realizado la segmentación de una imagen y justo antes de llamar al módulo *vecino8*.

3.4.5 Descripción del módulo de preprocesamiento *vecino8*

El módulo *vecino8* se encarga de tomar los datos que serán introducidos en la red neuronal, toma como entrada el resultado del proceso de ampliación que se encuentra almacenado en el archivo *letras2.txt*.

El objetivo del módulo es crear un vector de datos que contenga la media de pixeles en negro que se encuentran unidos a cada pixel. Para ello es necesario que el módulo verifique cual es el contenido de los pixeles que se encuentran alrededor de cada pixel en la imagen.

Como salida da el archivo *destino.txt*, que contiene los datos ser introducidos en la red neuronal, así como las etiquetas "espacio" e "Intro", las cuales solo tienen significado durante la etapa de funcionamiento normal y son interpretados por el módulo *identificador*.

Se hizo de este modo por el hecho de que los caracteres manuscritos difieren mucho en lo que a la posición de pixeles en un lugar se refiere, por tanto un mismo caracter podría parecer que tiene la misma cantidad de pixeles y en las mismas posiciones siempre, pero en la realidad varían ligeramente tanto en su cantidad como en su posición, dando resultados erróneos en el reconocimiento. Al sacar el valor promedio de un pixel en un área se minimiza el efecto de pixeles individuales para resaltar el de varios pixeles concentrados en un área, aunque se tiene el riesgo de que las variaciones entre distintas letras sean muy pequeñas tendiendo a confundir las letras, dado que al sacar el promedio de un área no se ha tomado en cuenta la ubicación relativa al pixel que se esta examinando (si los pixeles en negro encontrados se encuentran arriba, abajo o a los lados del pixel que esta siendo procesado), pero en las pruebas de reconocimiento este método fue el que presentó mejores resultados en general[17].

3.4.6 Programación del módulo *vecino8*

Para el módulo *vecino8* se tiene el diagrama de flujo mostrado en la figura 3.4.6.1.

Se lee la información en *letras2.txt*, si no es "espacio" o "Intro" se procede a sacar el valor del dato correspondiente al pixel que se esta examinando, en cualquiera de los dos casos el resultado es escrito en el archivo *destino.txt*.

Para el proceso de sacar la media se asume que un pixel con color negro posee un valor de 1, mientras para un pixel blanco es igual a cero, el resultado de sumar cada valor situado en el vecindario 8[18] y se agrega el valor del pixel que se esta examinando por tanto se tienen 9 datos, el valor de esa suma es dividido entre 9, representando que tantos pixeles de color negro se encuentran cerca del pixel que se esta examinando, un valor de 1 indica que tanto el pixel examinado como su vecindario 8 son de color negro, un valor de 0 por el contrario indica que ninguno de los pixeles del vecindario y el que se esta examinando es negro.

---

[17] Ver A.3 Resultados de las pruebas del ROCM.
[18] El concepto de Vecino 8 se explica en el apartado 3.4.2

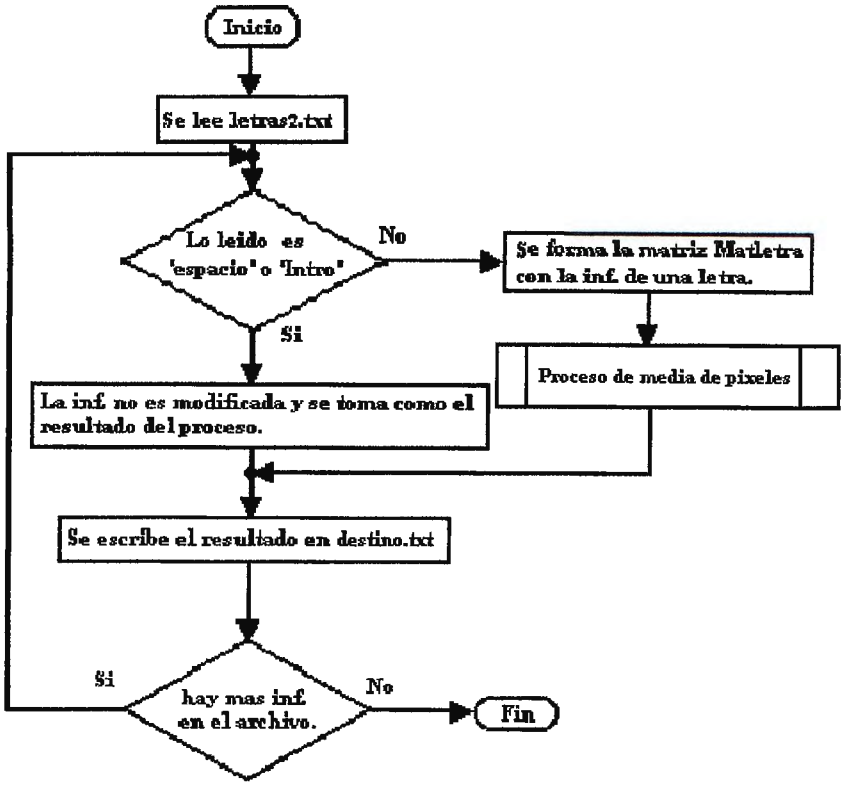


Figura 3.4.6.1. Diagrama de flujo para el proceso de *vecino8*, el resultado del proceso se guarda en forma de texto en el archivo *destino.txt*.

La fórmula que representa este proceso es la mostrada en la ecuación 3.4.2.1. Lo que se pretende resaltar con este método es que tantos pixeles se encuentran en negro alrededor del pixel que se esta examinando, es la cantidad media de pixeles en negro en el área del pixel.

3.4.6.1 Rutinas ocupadas en el módulo vecino8

Para este módulo solo se ocupa una rutina, la *tdcm*:

- **tdcm:** Es invocada justo después que *zoom*, pues trabaja con el resultado de esta última. Toma los datos contenidos en el archivo *destino.txt* y procede a verificar si es alguna de las etiquetas que no es tomada en cuenta para el proceso, que no son afectadas por *tdcm* y por tanto son directamente escritas en el archivo de salida o si por el contrario lo que se tiene son los datos de una letra. Para el segundo caso toma cada pixel y verifica su valor y el de los pixeles ubicados en su vecindario 8, los valores posibles son de 1 (negro) ó 0 (blanco), procediendo a sacar el promedio de los valores encontrados. Dicho promedio es la respuesta buscada y se escribe en el archivo *destino.txt*, por tanto presenta tantas respuestas por letra como pixeles posea dicha letra, los valores son escritos en una línea por cada letra, es decir que si el texto examinado tuviese dos letras el archivo *destino.txt* tendría solo dos líneas, sin tomar en cuenta las etiquetas "espacio" o "Intro". El proceso de la *tdcm* no se ve afectado si el programa se encuentra o no

creando un nuevo usuario, lo único que cambia es por quien es invocada esta rutina, si por el módulo *nusuario* (cuando se esta creando un nuevo usuario) o por el módulo *identificador* (si el programa esta trabajando normalmente). Esta función no necesita argumentos y da como resultado el archivo *destino.txt*.

3.5 IDENTIFICADOR DE CARACTERES.

3.5.1 Descripción del módulo para identificar caracteres

Cuando se ha ingresado como un usuario válido y se desea realizar la identificación de un texto, al presionar el botón *Procesar Imagen* en la pantalla del segmentador, figura 3.2.3, la *IGU* llama primero al módulo *segmentar* y posteriormente éste módulo llama al *identificador*, el cual se encarga de los pasos necesarios para mostrar en el cuadro de texto principal del programa los caracteres reconocidos.

El módulo *identificador* se encarga de mostrar el resultado de la segmentación para que el usuario pueda realizar algún cambio en los parámetros del segmentador, llamar al módulo *segmentar*, de ser necesario, y si el usuario acepta la segmentación se encarga de invocar al módulo *zoom* para ampliar la imagen segmentada, después invoca a *vecino8* y posteriormente es el *identificador* quien se encarga de tomar los resultados de *vecino8* para alimentar a la red neuronal para que esta haga el reconocimiento, el *identificador* toma la salida activa de la red neuronal y la transforma en una letra, esto gracias a que posee la relación existente entre las salidas de la red y la letra que representan, dicha letra es ingresada en la ventana de texto principal del programa y se continúa con el proceso hasta haber reconocido todas las letras segmentadas, además si aparecen las etiquetas "espacio" o "Intro" son ingresados en la ventana de texto principal pero con el caracter correspondiente (un espacio en blanco para el primero y un cambio de línea con retorno de carro para el segundo).

3.5.2 Programación del módulo *identificador*

Para el módulo *identificador* se tiene el diagrama de flujo mostrado en la figura 3.5.2.1. Este módulo es llamado por *segmentar*, y le pasa como argumento el nombre del usuario, que el módulo *segmentar* obtiene del programa principal.

Con el nombre del usuario se cargan los archivos de parámetros, relación entre las salidas de la red neuronal con las letras que representan y los pesos de la red.

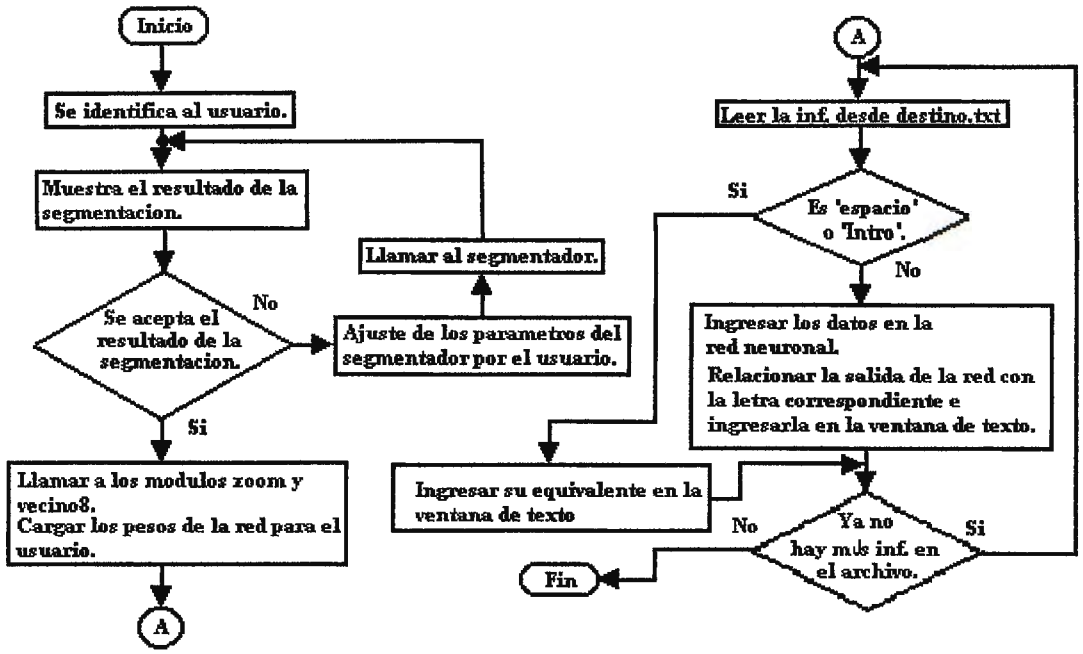


Figura 3.5.2.1. Diagrama de flujo para el módulo *identificador*.

Después carga el resultado de la segmentación que se encuentra en el archivo *letras.txt*, y es mostrado en la pantalla de manera similar a la etapa de entrenamiento, es decir que aparecen los caracteres segmentados en una ventana donde además se muestran 4 botones de comando ('Cancelar', 'Realizar otra iteración', 'Acepto el resultado' y 'Ayuda') con la diferencia de que ahora todos los botones están habilitados, el usuario puede aprobar la segmentación o realizar otra iteración para segmentar de nuevo la imagen pero con parámetros modificados, los ajustes que se realicen en los parámetros ahí mostrados también afectarán los que posee la IGU, y pueden ser guardados posteriormente en la configuración del usuario ya existente.

Si el usuario aprueba el resultado de la segmentación entonces llama al módulo *zoom* para ampliar y regularizar los resultados del segmentador, produciendo el archivo *letras2.txt*, después al módulo *vecino8* para crear los datos que serán ingresados en la red neuronal, estos datos son almacenados en el archivo *destino.txt*.

Los datos en *destino.txt* son tomados por el módulo *identificador*, se comprueban si son las etiquetas 'espacio' o 'Intro', si lo son no llama a la red neuronal y sus equivalentes son ingresados en la ventana de texto principal, si no son las etiquetas mencionadas entonces los datos pasan a la entrada de la red neuronal, la ART 2 al final del proceso regresa el número de la neurona de salida activada, el número de la neurona activada corresponde con la letra de texto que representa y cuya relación se guarda en el archivo de relación correspondiente del usuario. El caracter es insertado en la ventana de texto principal y si hay más datos repite el proceso de lectura del archivo *destino.txt* y la identificación hasta que ya no quede más información sin procesar.

3.5.2.1 Rutinas ocupadas en el módulo identificador

Las rutinas ocupadas por el módulo identificador son:

- **qLetrasEs:** Es llamado por el módulo *segmentar* y se encarga de identificar al usuario con el que se esta trabajando y mostrar el resultado de la segmentación. El resultado de una segmentación siempre se guarda en un archivo temporal que es accesado por esta función, el cual contiene las letras de la manera en que el programa las ha segmentado. Para poder representar lo contenido en el archivo temporal resultante de la segmentación esta función traduce los valores allí encontrados en los términos 'black' si el valor es 1 y 'white' si el valor es 0, dichos términos son comprendidos por el módulo TK y los traduce en cambios en el color de los pixeles de la ventana que muestra los resultados. Posee botones para incrementar y decrementar los valores de los tamaños de la ventana del segmentador y las tolerancias, ninguno de esos valores puede valer menos de 1, y si se ingresa algo menor es cambiado a 1. Si se ingresa un número decimal solo se toma la parte entera. Para realizar los decrementos e incrementos de los valores usando los botones de comando se llama a las funciones: *Txd, Txa, Tyd, Tya, Tolxd, Tolxa, Tolyd* y *Tolya*. Si se acepta el resultado se llama a la función *qLetraEs2* pasándole como argumento el nombre del usuario. Si se presiona 'Realizar otra iteración' llama al módulo *segmentar* para segmentar de nuevo la imagen con los parámetros actuales de segmentación que pudieron haber sido variados por el usuario.
- **qLetraEs2:** Recibe como argumento el nombre del usuario y se abre los archivos correspondientes a los parámetros, relación de letras y los pesos del usuario. Llama al módulo *zoom* para incrementar el tamaño de los caracteres y regularizar las letras, es decir, que todas tengan la misma anchura y altura, después llama al módulo *vecino8* para crear el archivo con los datos que servirán de entrada para la red neuronal. Inicializa la red ART 2, carga los pesos de la red para el usuario especificado, abre el archivo *destino.txt* creado por el módulo *vecino8*, lee una línea de datos y si el dato leído es 'espacio' o 'Intro' pone el caracter equivalente en la ventana de texto principal, sino se trata de una hilera de valores numéricos que serán ingresados en la red neuronal, se pasa esta hilera como argumento y se procesa por el módulo *art2*, el cual devuelve el número de la salida que se activa, con este resultado la función verifica a que letra se refiere al examinar el archivo relación del usuario, la letra identificada es ingresada en la ventana de texto principal, verifica si hay más información en el archivo *destino.txt*, si la hay repite el proceso de identificación desde la lectura de la información del archivo hasta la verificación de si hay todavía información sin procesar, de lo contrario termina el proceso del módulo *identificador*.
- **Txd:** Toma el valor existente en *$tx* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Txa:** Toma el valor existente en *$tx* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tyd:** Toma el valor existente en *$ty* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.

- **Tya:** Toma el valor existente en *$ty* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolxd:** Toma el valor existente en *$tolx* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolxa:** Toma el valor existente en *$tolx* lo aumenta en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolyd:** Toma el valor existente en *$toly* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.
- **Tolya:** Toma el valor existente en *$toly* lo decrementa en uno, toma su parte entera, verifica que sea mayor que uno de lo contrario asigna uno a esta variable y lo inserta en el control de entrada de texto correspondiente.

4. CONCLUSIONES

- Se pudo crear un programa que convierte la información escrita, contenida en una imagen, en texto que puede ser interpretado por cualquier procesador de texto de Windows. Pero posee un bajo porcentaje de reconocimiento para letra manuscrita y medio o alto grado de acierto para letra de máquina.

- Se comprobó que las redes neuronales pueden aplicarse para el reconocimiento de texto.

- Se logró que el programa fuera capaz de hacer que la red neuronal "aprendiera" la forma de escribir del usuario, es decir, es capaz de adaptarse a las diversas escrituras que poseen las personas, pero nunca de manera perfecta, siempre existen errores en el reconocimiento.

- Con la elaboración de este trabajo se mostró las ventajas que ofrece el lenguaje Perl, para poder ser una alternativa de programación frente a otros lenguajes utilizados tradicionalmente.

- Con las diversas pruebas hechas con la letra manuscrita se puede afirmar que es muy difícil crear un programa capaz de reconocer arriba del 90% de las letras contenidas en una imagen, y que el porcentaje de acierto depende en gran medida de la irregularidad con la que escribe el usuario.

- Se comprobó que la letra manuscrita es mucho más difícil de reconocer que la letra de máquina, por esa razón los OCR para letra manuscrita tienen un alto precio en el mercado (arriba de $500) y pocas compañías los producen.

- Con la letra manuscrita el porcentaje de efectividad se sitúa entre el 30 al 80%, dependiendo de situaciones como la regularidad de la escritura y la calidad de la imagen ingresada (poco ruido, resolución arriba de 150 dpi en la digitalización, que no aparezcan imágenes ajenas al texto).

- Los mejores promedios de reconocimiento se dieron para letra equiespaciada, la letra courier por ejemplo, que tuvo arriba del 80% de aciertos. Pero el porcentaje disminuye cuando hay ruido en la imagen digitalizada.

5. REFERENCIAS

1. Rivera, Ing. Eduardo, **Clases de la materia Sistemas de Control Automático II, ciclo 2 año 2001.**
2. Adobe productos, Adobe Capture disponible desde Internet en: http://www.adobe.com/products/acrcapture/main.html, última consulta 20/01/03.
3. Pixel Ware, Pixel File, disponible desde Internet en:: http://www.pixelware.com/file/index.html, última consulta 20/01/03.
4. Tratamiento de Imágenes, página principal disponible desde Internet en:: http://documents.cfar.umd.edu/, última consulta 5/01/03
5. ART (teoría de resonancia adaptativa), http://web.umr.edu/~tauritzd/art/welcome.html, última consulta 14/01/03.
6. Perl, Página principal disponible desde Internet en: http://www.ActiveState.com/, última consulta 20/05/03.
7. Perl, Active Perl User Guide, viene con el programa Perl.
8. Programming Perl, **Larry Wall,** second edition, O'Reill.
9. Hilera, José R., **Redes Neuronales artificiales, fundamentos, modelos y aplicaciones,** editorial Alfaomega.

A. ANEXOS

A.1 Ayuda del programa ROCM.

A continuación se muestra el contenido de la ayuda del programa:

A.1.1. ¿Qué es el reconocimiento óptico de caracteres?

Es la identificación de los caracteres captados por una imagen, por ejemplo por medio de un escáner o una cámara, y convertirlos en sus equivalentes de texto.

Los reconocedores ópticos de caracteres (ROC) se pueden clasificar en 4 categorías según su complejidad:

1- ROC de caracteres tipográficos (caracteres impresos por alguna máquina).

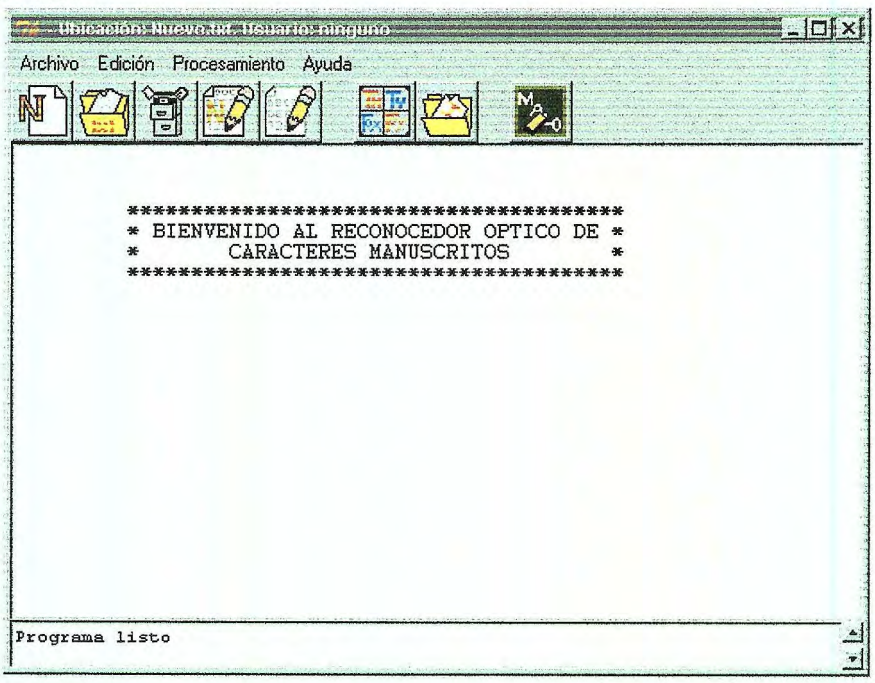
2- ROC de caracteres escritos a mano equiespaciados (como los escritos en las casillas de los formularios).

3- ROC de caracteres escritos a mano con espacio entre caracteres (letra de molde).

4- ROC de caracteres escritos a mano sin espacio entre caracteres (letra de carta).

Este programa esta diseñado para trabajar entre los tipos 3 y 4, aunque, si se varian los parámetros de segmentación apropiadamente y se entrena adecuadamente, puede trabajar para los tipos 1 y 2.

La clave esta en tratar de mantener lo más constante posible el tamaño de la letra (ancho y alto) así como los espacios. Entre más regular sea la letra mejor será el reconocimiento.



A.1.2. Generalidades acerca del programa ROCM - M.

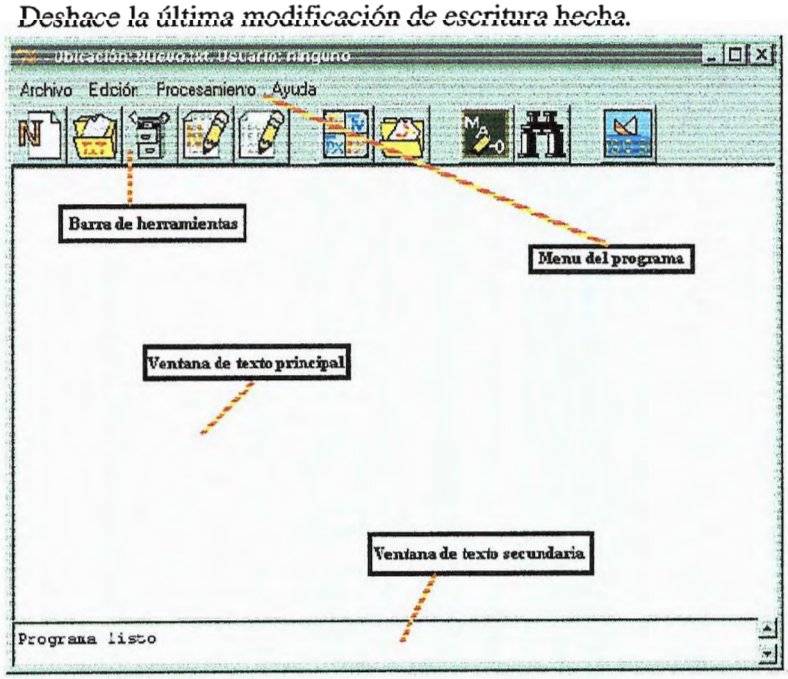
El programa se encarga de renocer los caracters manuscritos de una imagen previamente obtenida por algún digitalizador, tal como un escáner o una cámara, puede reconocer archivos de imagen de extensión .bmp o .gif, el resultado del reconocimiento es un archivo de texto simple (.txt) que es guardado en la computadora.

El ROCM - M tiene la capacidad de ser personalizado por el usuario para mejorar la calidad de la identificación, esto se logra por medio de la personalización de la red neuronal que es la encargada del reconocimiento. Cada usuario puede crear archivos de parámetros personalizados usando el creador de nuevos usuarios que posee el programa.

TECLAS DE ACCESO DIRECTO:

Hay ciertas funciones que pueden ser activadas desde el teclado, las funciones definidas para el programa y su combinación de teclas se muestra a continuación:

Alt + a Abre un archivo de texto.
Alt + b Borra el contenido de la ventana principal.
Alt + c Crea un nuevo usuario
Alt + g Guarda el contenido de la ventana como un archivo de texto.
Alt + n Crea un nuevo archivo de texto.
Alt + r Reemplazar texto
Alt + u Para llamar los parámetros de un usuario ya existente.
Ctrl + c Copia el texto seleccionado.
Ctrl + v Para pegar el texto copiado o cortado.
Ctrl + x Corta el texto seleccionado.
Ctrl + z *Deshace la última modificación de escritura hecha.*

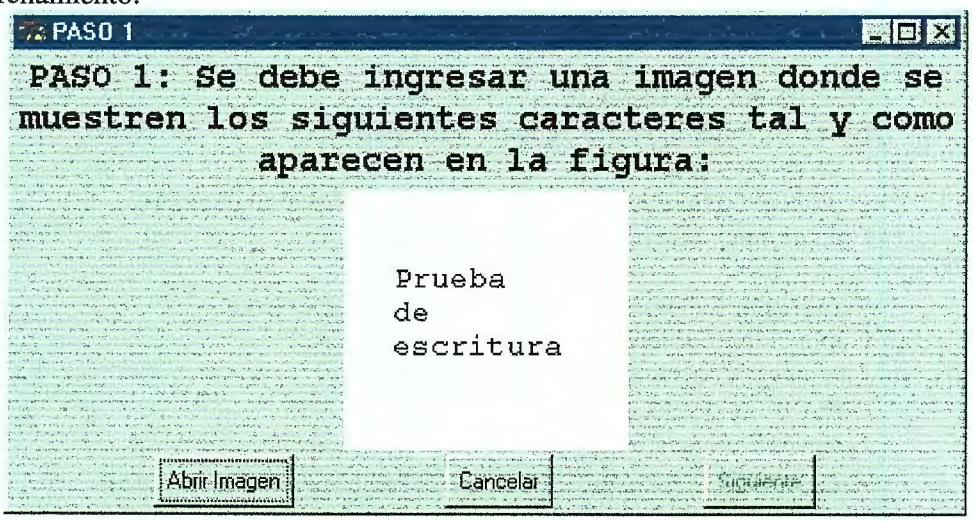


A.1.3. Creación de nuevos usuarios y entrenamiento de la red.

El programa ROCM - M crea usuarios para mejorar la calidad del reconocimiento. Para la creación de un usuario es necesario entrenar la red y contar con imágenes que contengan las letras que se solicitan por el asistente para la creación de nuevos usuarios (-Archivo->-Creación de Nuevos Usuarios), con esas imágenes se ajustarán los parámetros de la red neuronal (que es la encargada de la identifcación) a la escritura del usuario.

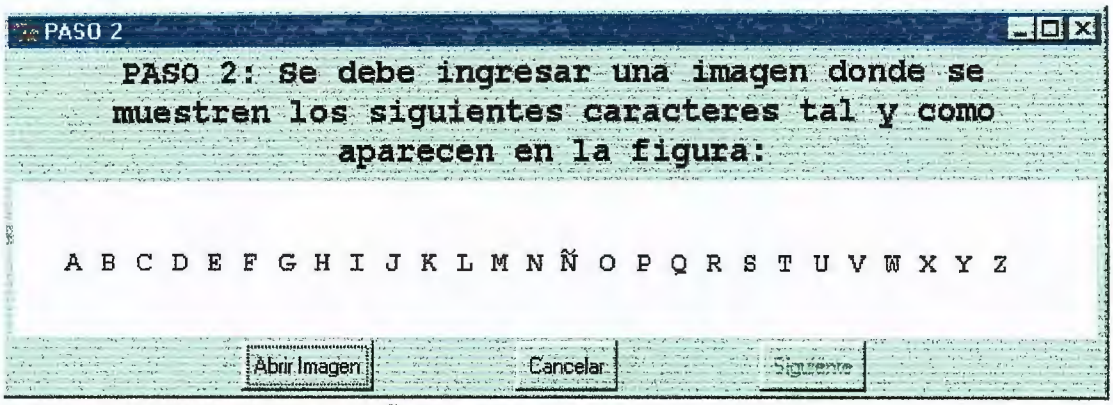
Para el entrenamiento de la red es necesario contar con 4 imágenes en formato bmp o gif (las imágenes deben venir en blanco y negro o no se asegura un buen reconocimiento), las cuales deberán contener las siguientes letras o símbolos escritos a mano por el usuario espaciados debidamente:

Imagen 1: Se muestra la pantalla de petición de la imagen a trabajar tal y como aparece en el entrenamiento:



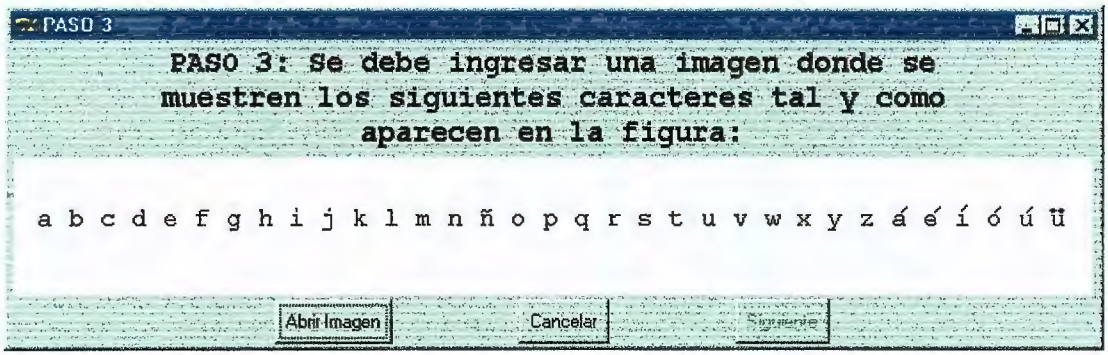
Prueba
de
escritura

Imagen 2: Se muestra la pantalla de petición de la imagen a trabajar tal y como aparece en el entrenamiento:



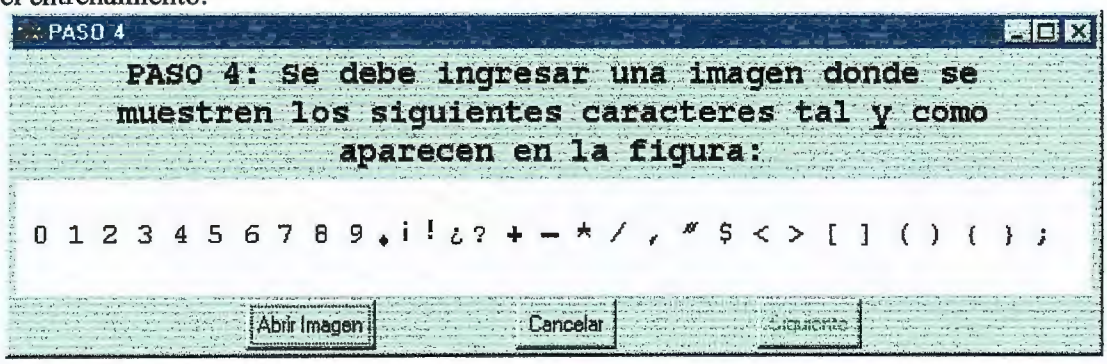
A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z

Imagen 3: Se muestra la pantalla de petición de la imagen a trabajar tal y como aparece en el entrenamiento:



a b c d e f g h i j k l m n ñ o p q r s t u v w x y z á é í ó ú ü

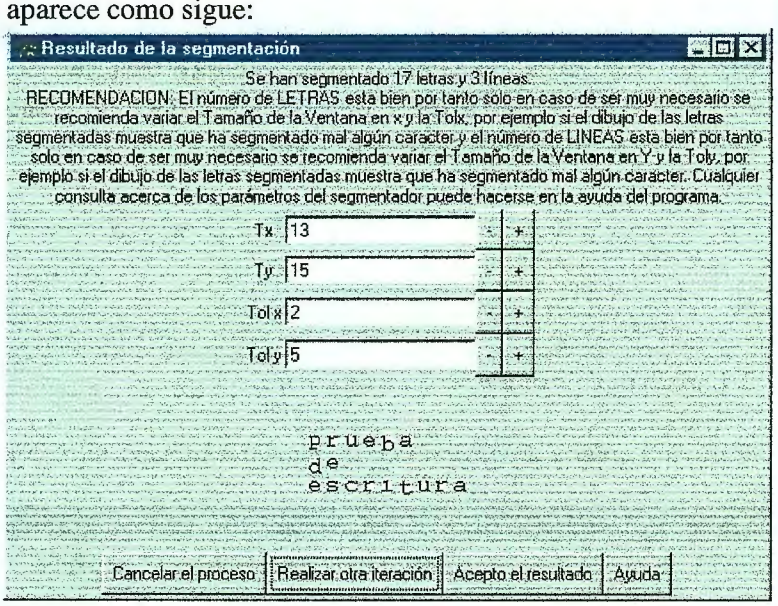
Imagen 4: Se muestra la pantalla de petición de la imagen a trabajar tal y como aparece en el entrenamiento:



0 1 2 3 4 5 6 7 8 9 . ¡ ! ¿ ? + - * / , " $ < > [ ] ( ) { } ;

Cuando se escoge una imagen, al presionar "Abrir Imagen" de la ventana del Paso, aparece la imagen junto con dos botones, como se muestra en la figura sig:

Al presionar el botón Procesar la imagen es segmentada, tarda cierto tiempo dependiendo del tamaño de la imagen y la velocidad de la máquina, el resultado de la segmentación aparece como sigue:



Las imágenes de los pasos abarcan todas las letras o símbolos que el programa es capaz de reconocer, si ocurre algún error durante el entrenamiento, por ejemplo el programa notifica que hay menos caracteres que los previstos, verifique que las imágenes ingresadas contienen los caracteres pedidos así como de que exista un espacio entre caracter y caracter para las letras que aparecen en los pasos 2, 3 y 4, si aún así no funciona cree otra imagen con los caracteres solicitados con el espacio adecuado.

Esto es válido tanto si desea reconocer letras de máquina o manuscritas.

ACERCA DE LOS PARÁMETROS DEL SEGMENTADOR:

El segmentador es la parte del programa que se encarga de tomar la imagen y dividirla en regiones cuyo tamaño no sobrepase lo dado por los parámetros Tx (ancho

máximo del caracter en pixeles, su valor esta guardado en la variable $tx), Ty (altura máxima del caracter en pixeles, almacenado en la variable $ty) y Tolerancia (que tan lejos puede estar un pixel y aún se toma como perteneciente a un caracter, la distancia también esta dada en pixeles, valor almacenado en $tolx y $toly). En dichas regiones es donde se presume hay un caracter, y lo primero que hace es encontrar es la primera y última fila con información (pixeles en negro) y la primera y última columna con información (pixeles en negro), después de eso explora de arriba hacia abajo de izquierda a derecha hasta encontrar un pixel en negro desde comienza a crear una pequeña matriz, llamada ventana, en la cual se ponen todos los pixeles que aparecen conectados al primer pixel o a una distancia de separación máxima dada por el parámetro de Tolerancia ($tolx en la dir horizontal y $toly en la dir vertical), cuando ya se ha alcanzado los valores de Ty y Tx imprime el caracter encontrado en un archivo de texto, con unos y ceros, y se pasa a buscar otro pixel y se repite el proceso hasta llegar al final de la imagen.

Para saber si un pixel esta conectado se evalúa si dicho pixel esta dentro del rango de tolerancia dado, yendo un número de pixeles desde la izquierda hacia la derecha, se baja un pixel y se prueba otra vez de izquierda a derecha y se vuelve a bajar, se repite el proceso hasta encontrar un pixel o alcanzar los valores límites del programa.

TX: Indica la anchura máxima de pixeles conectados que admite el programa como parte de una letra. Si la letra es más ancha será cortada en el valor TX.
TY: Indica la altura máxima de pixeles conectados que admite el programa como parte de una letra. Si la letra es más alta será cortada en el valor TY.
TOLX: indica la distancia en pixeles que puede estar alejado, en la dirección HORIZONTAL, un pixel en negro de otro y ser considerado como parte del mismo caracter.
TOLY: indica la distancia en pixeles que puede estar alejado, en la dirección VERTICAL, un pixel en negro de otro y ser considerado como parte del mismo caracter.

RECOMENDACIONES PARA EL ENTRENAMIENTO:

Si los caracteres no son segmentados de forma correcta debe verificarse en la ventana que muestra los resultados de la segmentación para variar el parámetro adecuado:

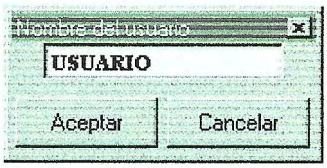
PROBLEMA	RECOMENDACIÓN
- Falta la parte superior de las letras.	- Aumentar TY puede que la altura sea muy grande.
- Falta la parte lateral de las letras.	- Aumentar TX puede que la anchura sea muy grande.
- TX y/o TY se han variado y no toma todos los pixeles pertenecientes a una letra.	- Aumentar TOLX y/o TOLY según sea el caso.

¡IMPORTANTE! ¡IMPORTANTE! ¡IMPORTANTE! ¡IMPORTANTE!

Al realizarse los pasos verifique en la ventana de segmentación la forma en que han sido segmentados los caracteres. Cuide que tanto las letras segmentadas como el orden en que aparecen sea el mismo que el de la imagen mostrada en el paso que se ha concluido, ya que el programa tomará esas formas para reconocer las letras que se le ingresarán posteriormente y si, por ejemplo, durante la segmentación aparece una línea vertical en donde debería aparecer la letra "A" entonces al estar corriendo el programa y aparece una línea vertical la tomará como una letra "A" y así aparecerá en el texto de respuesta.

A.1.4. Usuarios ya creados (Como ingresar y realizar un reconocimiento) definición de los pesos de la red.

Los usuarios ya existentes en el programa pueden abrir su configuración por medio de:
Archivo->Usuario
o presionando Alt + U desde el teclado. Aparecerá un cuadro donde escribirán su identificador (nombre de usuario) y el programa cargará los parámetros establecidos para dicho usuario. Dicho cuadro es el siguiente:



Si desea trabajar como otro usuario ya existente entonces se debe acceder de nuevo a Usuario y escribir el nuevo identificador (nombre de usuario).

Al realizar este proceso se podrá realizar una sesión de reconocimiento y el programa usará los parámetros de segmentación y los pesos de red que corresponden al usuario que en ese momento se ha accedido.

El programa OCR viene con dos usuarios predefinidos uno para letra manuscrita de molde y otra para letra de máquina como la que se encuentra comunmente en los libros. Estos usuarios pueden ser ocupados por cualquiera y se recomienda su uso para familiarizarse con el programa. Los usuarios son:

- ninguno — Es el usuario por defecto y es para letra de molde
- maquina — Para letra tipo imprenta.

Después de escoger el usuario con el que se va a trabajar es necesario ingresar la imagen que va a ser reconocida, esto se puede hacer por:
Procesamiento->Imagen a procesar
o presionando el botón en la barra de herramientas para "Abrir Imagen para procesar", que se muestra en la siguiente figura:

Se abre un cuadro de diálogo para ingresar la imagen (tipo bmp o gif) que contiene la inf. Cuando se ha escogido la imagen es desplegada en la ventana para iniciar la segmentación donde aparecen los botones para Procesar la imagen y Cancelar el proceso.

Al presionar el botón "Procesar" se inicia el proceso de reconocimiento, y al final de proceso se muestra el resultado en la ventana de texto principal, además muestra un cuadro de diálogo para saber si se desea grabar los pesos resultantes del reconocimiento, esto debido a que cada vez que se realiza un reconocimiento los pesos de la red varían para adapatarse los caracteres con los que fue entrenada con la forma de los patrones que se han reconocido, ES RECOMENDABLE GRABAR LOS PESOS* EN CADA RECONOCIMIENTO SI LA LETRA DEL USUARIO ES VARIABLE.

El resultado del reconocimiento es puesto en la ventana de texto principal para que el usuario pueda modificarlo a su antojo, para lo cual el programa brinda la herramienta de "Reemplazar el texto" y la "Corrección automática", esta última es aplicable si el texto tiene sentido para el idioma español, realiza correcciones a problemas algo comunes, como la confusión de la ' e' con la ' c' , y la aparición de mayúsculas cuando son minúsculas. Se puede acceder a esta función así:
Procesamiento->Corrección Automática
o presionando el botón en la barra de herramientas para "Corrección automática", que tiene la siguiente imagen:

En cualquier momento después del reconocimiento se puede guardar el texto resultante en un archivo de texto con:
Archivo->Guardar archivo
o presionando el botón de la barra de herramientas para "Guardar archivo de texto" cuya imagen es la siguiente:

También se pueden realizar búsquedas y reemplazos con la función ' Reemplazar' , que puede ser accedida por:
Edición->Reemplazar
o presionando el botón de la barra de herramientas para "Reemplazar texto", que tiene la siguiente imagen:

De cualquier manera aparecerá una ventana donde se deben ingresar el texto a buscar y su reemplazo, si se presiona "Buscar siguiente" cuando encuentre una coicidencia preguntará si desea que sea reemplazada, si se presiona "Reemplazar todos" sustituirá todas las coincidencias sin preguntar.

* Los pesos son los elementos de la red neuronal que le sirven para recordar la forma de las letras con las que se ha entrenado, la red usada para este reconocedor de caracteres tiene la característica de que puede variar su pesos al momento de haber reconocido un caracter para ajustarlo a la entrada con la que fue activada, si las letras han sido reconocidas correctamente se recomienda grabar los pesos ya que la red tiene sus pesos ajustados para

la letra ingresada, si las letras reconocidas es menor al 80% es mejor no grabar los pesos, si el reconocimiento es pobre se recomienda crear un nuevo usuario.

A.1.5. Reconocimiento no satisfactorio.

Cuando el reconocimiento es de baja calidad pruebe a ajustar los siguientes parámetros del segmentador (Segmentador->Parámetros del segmentador):

Tx: ajusta el ancho de la ventana que pasa encima de los caracteres, un valor grande es apropiado para letras anchas.

Ty: ajusta el alto de la ventana que pasa encima de los caracteres, un valor grande es apropiado para letras muy altas.

Tolx y Toly: ajusta el valor de tolerancia que ocupa el segmentador para identificar si un pixel pertenece o esta unido a una imagen, se recomienda no variar mucho este valor si no ha probado con los anteriores.

Consejos adicionales:

- Algo que debe tomar en cuenta es que cualquier imagen será procesada por el programa como si se tratase de una letra, por tanto se debe cuidar al momento de realizar la digitalización de la imagen que solo se aprecie la información correspondiente a las letras, líneas de cuaderno pueden ser obviadas por los escaners, generalemente escogiendo la opción para digitalizar en blanco y negro, o con la opción OCR.

- Si nota que aparecen muchos menos caracteres de los que existen en la imagen, aunque los caracteres no sean identificados correctamente deberían aparecer representados por otros caracteres que indica que no ha podido reconocer el caracter, disminuya los valores de Tx y Ty, y en último término el de las tolerancias.

- Si la segmentación esta bien el problema es en la red por lo que habría que revisar si las letras difieren mucho de las usadas durante el entrenamiento por lo que debería crearse un nuevo usuario para ese tipo de letra.

- Si aparecen más caracteres de los existentes en la imagen aumente el valor de los parámetros.

- Revisar si en la imagen ingresada pueden apreciarse correctamente los caracteres.

A.1.6. Límites del programa.

- No se asegura el buen reconocimiento de texto con un tipo de letra distinto al ocupado para el entrenamiento.

- Las imágenes que se van a trabajar deben venir en dos colores, blanco(fondo) y negro (texto).

- El programa no se encarga de obtener la imagen del digitalizador.

- En caso de baja calidad en el reconocimiento consulte la sección 5 de la ayuda para ajustar los parámetros.

- El tiempo que se toma en reconocer una imagen depende del tamaño de ésta y de las capacidades del sistema.

- No es capaz de reconocer fórmulas matemáticas o tablas.

- Elimine todos los caracteres extraños tales como líneas, rayas etc. que pueden ser mal interpretados por el programa.

- Esta pensado para usarse en el idioma español y sólo reconocerá los caracteres que aparecen en el entrenamiento.

- No se conservarán tamaños de fuente ni el formato del documento, excepto por los espacios y cambios de línea.

- El archivo de salida será de solo texto.

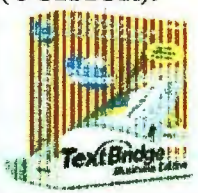
A.1.7. Acerca de este programa.

Este programa fue creado por:

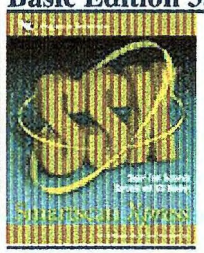
Miguel Eduardo Flores Gómez

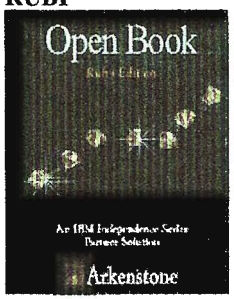
como un trabajo de graduación para la carrera de Ing. Electrónica el año 2004. El Salvador, C. A.

A.2 Tabla comparativa de OCR existentes en el mercado

Producto	Fabricante	Idiomas	Formatos de entrada	Formato de salida	Requisitos del sistema.	Precio sin IVA
Omnipage professional 11 para Windows. Castellano (OCR/ICR). 	Scansoft	Reconoce 100 idiomas, incluso cuando se tiene varios idiomas en la misma página.	♣ PDF ♣ Documentos impresos en digitales ♣ Tiff-FX ♣ PaperPort Max cuando utiliza Paperport	♣ PDF ♣ HTML WYSIWYG ♣ Escanea directamente a cualquier editor de textos, compatible con Microsoft Office 2000	CD-ROM: Si Disco duro: 140 MB Memoria RAM: 32 MB (se recomiendan 64 MB) Procesador: Intel Pentium o equivalente Sistema operativo: Windows 95, 98, 2000, ME y NT 4.0 o XP Tarjeta gráfica: Monitor SVGA, 800 x 600 como mínimo (256 colores o más) Hardware adecuado para la captura de imágenes.	$539,21
TextBridge Pro 9.0 W95/98/NT. Castellano (OCR/ICR). 	Scansoft	Compatible con 56 idiomas	Documentos impresos en digitales	♣ Escanea directamente a cualquier editor de textos, compatible con Microsoft Office 2000 ♣ Convierte documentos en páginas Web (HTLM).	Windows 95/98/2000/NT 4.0 PC 486 o superior 24 Mb RAM (32 recomendados) 20 Mb espacio libre en disco duro Hardware adecuado para la captura de imágenes.	$70,94
Omnipage	Scansoft	Comprende	Convierte	♣ Documento	Power Macintosh o superior, no	$94,23

Profesional 8.0 Macintosh Inglés (OCR/ICR) 		once idiomas.	documentos de papel digitalizados.	s de ordenador ♣ Salida HTML.	funciona en Macintosh de 68K Sistema 7.5 o superior 10 MB de RAM 25 MB de espacio en disco duro 640x480 pixel de resolución o superior Hardware adecuado para la captura de imágenes.	
Autodesk Raster Design Integración de imágenes raster y edición en AutoCAD (ICR)	Autodesk	–	♣ Reconoce textos manuscritos ♣ Mecanografiados en imágenes ráster ♣ Archivos de diseño DWG	♣ Archivos de diseño DWG ♣ Creación de archivos DWF ♣ Documentos en paginas Web.	AutoCAD 2000 de preferencia y los requisitos de sistema necesarios para este programa, Windows 98/2000/NT 4.0, 24 Mb RAM (32 recomendados) y 500 Mb libres. Hardware adecuado para la captura de imágenes.	$1.850
ABBYY FineReader 6.0 (OCR)	ABBYY	Reconoce 176 lenguajes, incluso combi-nados en una pagina. Dentro de estos 176 están 6 lenguajes de programación, formulas	Documentos impresos en digitales	♣ PDF ♣ HTML ♣ RTF ♣ Formato TXT	110 Mb de espacio libre en disco. Puerto USB para hardware protection key Windows 95 (OSR2 with USB-support) /98/ME/2000. Kofax Ascent Capture 5.0 instalado. Se recomienda: Intel Pentium II o mayor, Windows 2000, 128 Mb de memoria.	$129

		químicas simples y 4 lenguas artificiales.				
PROVEC A4 Estudio 3.5 Blanco y Negro. (OCR)	Wilzur Corp	–	Formatos PCX, RLC, Intergraph RLE y CIT, TIFF (sin comprimir, Packbits, Group 3 y 4), cualquier formato CAD o SIG.	Formatos AutoCAD (DXF) ASCII y Binary, MicroStation DGN, ARC/INFO Generate Format, PROGIS WINGIS (ADF), MapInfo (MIF/MID).	Computadora personal usando procesador 486 o superior. 16 Megabytes de RAM Sistema Operativo Microsoft Windows® 95, Windows® 98, Windows NT v.3.5x o v.4.0. 10 MB de Espacio en Disco para la instalación. Mouse compatible con Microsoft. Monitor VGA. Puerto Paralelo. Acceso a una unidad de disquette de 3.5" de alta densidad	$600 $700 (inc. manual)
Smartscan Xpress ICR Basic Edition 3.0 	Pegasus	–	♣ Manuscrito en mayúsculas, minúsculas y numérico. ♣ Mecanografiado en mayúsculas, minúsculas y numérico. ♣ Marcas de caracteres	Escanea directamente a cualquier editor de textos	Intel Pentium II 300 Mhz	$1201
Eyes & Hands FORMS 5	ReadSoft	Capaz de interpretar	Datos escritos a mano,			

ICR / OCR / OMR / código de barras 		información en unos 20 idiomas	impresos, códigos de barras, marcas, círculos, tablas, matrices, hasta direcciones completas.			
OPEN BOOK: RUBI 	Arkenstone	Reconoce siete idiomas, incluye interpretación de voz.	Acepta documentos con caracteres impresos de cualquier tipo y en contraste (cartas, faxes, revistas, libros, periódicos, etc.).	Escanea directamente a cualquier editor de textos	Pentium PC, 32Mb RAM, Windows 95, 98, NT4.0, 100 MB libres de disco duro, CD-ROM y teclado Windows	–

A.3 Resultados de las pruebas del ROCM

1- Pruebas realizadas para letra de máquina y manuscrita, para este caso se ocupaba la segmentación automática, no se realizaron ajustes manuales en los parámetros del segmentador, sino que quedaban con los resultantes del entrenamiento. Se usa la TDC en 2 dimensiones.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	41	21	51.2%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.8.
Courier	41	20	48.8%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 1.0.
Courier	41	17	41.5%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.6.
Courier	41	17	41.5%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.4.
Arial	58	24	41.4%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.6.
Italic	58	20	34.5%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.6.
Manuscrita	33	2	6.1%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 1.0.
Manuscrita	33	3	9.1%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.8.
Manuscrita	33	2	6.1%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.6.
Manuscrita	33	2	6.1%	4	4	100.0%	Primeras pruebas de reconocimiento con un rho de 0.4.

Tabla A.3.1. Para las letras regulares como la courier es mejor poner un factor de Rho bastante alto pero para la letra manuscrita esto no es posible, ya que si el rho es demasiado estricto entonces aumentan las letras mal reconocidas. Pero aún así el porcentaje de reconocimiento para la letra manuscrita es muy bajo.

2- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones. Se presentan los mejores promedios para cada tipo de letra

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	15	75.0%	4	4	100.0%	Rho = 0.6.
Arial	20	11	55.0%	4	4	100.0%	Rho = 0.6.
Italic	20	14	70.0%	4	4	100.0%	Rho = 0.6.
Manuscrita	33	4	12.1%	4	4	100.0%	Rho de 0.4.

Tabla A.3.2. El porcentaje de reconocimiento de letras manuscritas sigue siendo muy bajo, pero sigue presentando los mejores resultados con un rho bajo.

3- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. No se usa la TDC en 2 dimensiones. Se presentan solo los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	15	75.0%	4	4	100.0%	Rho = 0.6.
Arial	20	11	55.0%	4	4	100.0%	Rho = 0.4.
Italic	20	18	90.0%	4	4	100.0%	Rho = 0.6.
Manuscrita	20	2	10.0%	4	4	100.0%	Rho = 0.4.

Tabla A.3.3. La letra manuscrita se ve afectada de manera negativa al suprimir la TDC.

4- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones pero se añade un factor para aumentar el valor de los coeficientes pertenencientes a las frecuencias más bajas. Se presentan solo los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	13	65.0%	4	4	100.0%	Rho = 0.5.
Arial	20	12	60.0%	4	4	100.0%	Rho = 0.4.
Italic	20	15	75.0%	4	4	100.0%	Rho = 0.6.
Manuscrita	20	1	5.0%	4	4	100.0%	Rho = 0.4.

Tabla A.3.4. El poner el factor en la TDC afecta de manera negativa el reconocimiento en general.

5- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones y se aumentan la cantidad de coeficientes utilizados hasta ser el 30% de ($tx*$ty) antes era solo un 20%, se regularizan los tamaños de los caracteres segmentados Se presentan solo los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	17	85.0%	4	4	100.0%	Rho = 0.4.
Arial	20	13	65.0%	4	4	100.0%	Rho = 0.4.
Italic	20	18	90.0%	4	4	100.0%	Rho = 0.4.
Manuscrita	20	3	15.0%	4	4	100.0%	Rho = 0.4.

Tabla A.3.5. Hubo una ligera mejora en el reconocimiento de las letras en general.

6- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones y se aumentan la cantidad de coeficientes utilizados hasta ser el 60% de ($tx*$ty), tamaños regularizados de los caracteres segmentados. Se presentan los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	16	80.0%	4	4	100.0%	Rho = 0.4.
Arial	20	13	65.0%	4	4	100.0%	Rho = 0.4.
Italic	20	19	95.0%	4	4	100.0%	Rho = 0.4.
Manuscrita	20	3	15.0%	4	4	100.0%	Rho = 0.4.

Tabla A.3.6. No hubieron mejoras en general al aumentar el número de coeficientes.

7- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones y se aumentan la cantidad de coeficientes utilizados hasta ser el 100% de ($tx*$ty), tamaños regularizados de los caracteres segmentados. Se presentan los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	15	75.0%	4	4	100.0%	Rho = 0.4.
Arial	20	10	50.0%	4	4	100.0%	Rho = 0.4.
Italic	20	15	75.0%	4	4	100.0%	Rho = 0.4.
Manuscrita	20	1	5.0%	4	4	100.0%	Rho = 0.4.

Tabla A.3.7. Al aumentar al máximo de los coeficientes la red se volvió más estricta con ciertas letras y casi siempre que se equivocaba ponía símbolos o números en lugar de las letras que se esperaban, por tanto la red daba los resultados más desfavorables cuando no reconocía correctamente las letras, ya que ponía caracteres que poco o nada se parecían a lo buscado.

8- Para este caso se ocupaba la segmentación automática en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones cantidad de coeficientes 30% de ($tx*$ty), tamaños regularizados de los caracteres segmentados se variaron los parámetros de la red (a,b,c,e). Se presentan los mejores promedios.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Courier	20	17	85.0%	4	4	100.0%	Rho = 0.4, no hubo variaciones al variar los parámetros del segmentador.
Arial	20	13	65.0%	4	4	100.0%	Rho = 0.4, no hubo variaciones al variar los parámetros del segmentador.
Italic	20	18	90.0%	4	4	100.0%	Rho = 0.4, no hubo variaciones al variar los parámetros del segmentador.
Manuscrita	20	3	15.0%	4	4	100.0%	Rho = 0.4, no hubo variaciones al variar los parámetros del segmentador.

Tabla A.3.8. Al ir variando los parámetros a,b,c,e no hubieron grandes diferencias entre los resultados, el parámetro que más afecta a la red es el rho.

9- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones. Como la letra manuscrita es la que presenta los peores resultados esta prueba esta enfocada a ella. Se varían la cantidad de coeficientes (n) y se colocan en distintos valores los datos agregados en la segmentación para regularizar el tamaño de la imagen (0,-1 o 0.5).

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	1	5.9%	3	3	100.0%	n=0.1, rho=0.5, caracteres agregados=0
Manuscrita	17	5	29.4%	3	3	100.0%	n=0.3, rho=0.5, caracteres agregados=0
Manuscrita	17	5	29.4%	3	3	100.0%	n=1.0, rho=0.5, caracteres agregados=0
Manuscrita	17	5	29.4%	3	3	100.0%	n=0.5, rho=0.5, caracteres agregados=0
Manuscrita	17	3	17.6%	3	3	100.0%	n=0.3, rho=0.5, caracteres agregados=-1
Manuscrita	17	2	11.8%	3	3	100.0%	n=0.3, rho=0.5, caracteres agregados=0.5
Manuscrita	17	0	0.0%	3	3	100.0%	n=1.0, rho=0.5, caracteres agregados=-1

Tabla A.3.9. Al momento de convertir las distintas imágenes segmentadas al mismo tamaño se tiene que agregar algún dato para llenar ese espacio, se escogieron distintos valores pero con el presentaba mejores resultados era al agregar ceros.

10- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC en 2 dimensiones. La cantidad de coeficientes se fija al 30% de $tx*$ty, se agrega un método para duplicar la inf de la imagen segmentada es los datos que se agregan para regularizar el tamaño.

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	4	23.5%	3	3	100.0%	rho=0.5
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.6
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.7
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.8
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.9
Manuscrita	17	1	5.9%	3	3	100.0%	rho=1.0

Tabla A.3.10. Al repetir la información de la imagen en los datos que se agregan para regularizar el tamaño se aprecia que a veces se forman letras parecidas a otras, y entre más grande sea la diferencia de tamaño entre lo segmentado y lo que se busca es peor el reconocimiento.

11- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. No se usa la TDC. La cantidad de coeficientes se fija al 30% de $tx*$ty, se usa otra red neuronal para el preprocesamiento de los datos (Hopfield, heteroasociativa, ART1).

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	1	5.9%	3	3	100.0%	Hopfield, rho de la ART=0.3
Manuscrita	17	0	0.0%	3	3	100.0%	Hopfield, rho de la ART=0.5
Manuscrita	17	1	5.9%	3	3	100.0%	Hopfield, rho de la ART=1.0
Manuscrita	17	1	5.9%	3	3	100.0%	Heteroasociativa, rho de la ART=0.3
Manuscrita	17	2	11.8%	3	3	100.0%	Heteroasociativa, rho de la ART=0.5
Manuscrita	17	1	5.9%	3	3	100.0%	Heteroasociativa, rho de la ART=1.0

Tabla A.3.11. Se uso una red en lugar de la TDC como preprocesamiento de la imagen, ya que las dos redes probadas han sido ocupadas anteriormente para reconocimiento de imágenes, pero los resultados fueron muy malos en todas las pruebas hechas.

12- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. No se usa la TDC. La cantidad de coeficientes se fija al 100% de $tx*$ty, se usa la ART1 en lugar de la ART2.

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.6
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7
Manuscrita	17	1	5.9%	3	3	100.0%	rho=0.8
Manuscrita	17	1	5.9%	3	3	100.0%	rho=0.9
Manuscrita	17	0	0.0%	3	3	100.0%	rho=1.0

Tabla A.3.12. Se utilizó la red ART1 debido a que en este caso la imagen no es transformada sino que presenta datos binarios, por tanto se puede ocupar esta red. Los resultados muestran que no realizó un mejor reconocimiento que la ART2.

13- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se usa la TDC. La cantidad de coeficientes se fija al 30% de $tx*$ty, se usa la ART2. Se omiten los valores negativos procedentes de la TDC.

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.5
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.6
Manuscrita	17	1	5.9%	3	3	100.0%	rho=0.7
Manuscrita	17	1	5.9%	3	3	100.0%	rho=0.8
Manuscrita	17	0	0.0%	3	3	100.0%	rho=0.9
Manuscrita	17	0	0.0%	3	3	100.0%	rho=1.0

Tabla A.3.13. Se comprobó que no se pueden obviar las salidas negativas de la TDC, pues al omitirlos se pierde información valiosa para el reconocimiento.

14- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se utilizan diversas máscaras en los datos segmentados para resaltar características como bordes, o engrosar la letra. Se usa la TDC. La cantidad de coeficientes se fija al 30% de $tx*$ty, se usa la ART2.

	LETRAS			LINEAS			
Tipo de Letra:	En el texto	Reconocidas correctamente	% de acierto	En el texto	Reconocidas correctamente	% de aciertos	OBSERVACIONES
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7, resalta bordes
Manuscrita	17	1	5.9%	3	3	100.0%	rho=1.0, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.5, media de los pixeles
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7, media de los pixeles
Manuscrita	17	1	5.9%	3	3	100.0%	rho=1.0, media de los pixeles

Tabla A.3.14. El uso de las máscaras no afecto en gran medida el funcionamiento del programa.

15- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se amplian los datos segmentados hasta llegar al tamaño dado por $tx y $ty. Se usa la TDC. La cantidad de coeficientes se fija al 30% de $tx*$ty, se usa la ART2.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7, resalta bordes
Manuscrita	17	1	5.9%	3	3	100.0%	rho=1.0, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.5, media de los pixeles
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7, media de los pixeles
Manuscrita	17	1	5.9%	3	3	100.0%	rho=1.0, media de los pixeles

Tabla A.3.15. El uso de las máscaras no afecto en gran medida el funcionamiento del programa.

16- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se amplian los datos segmentados hasta llegar al tamaño dado por $tx y $ty. No se usa la TDC

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=0.7, resalta bordes
Manuscrita	17	2	11.8%	3	3	100.0%	rho=1.0, resalta bordes
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5, media de los pixeles
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.7, media de los pixeles
Manuscrita	17	2	11.8%	3	3	100.0%	rho=1.0, media de los pixeles

Tabla A.3.16. El uso de las máscaras y no usar la TDC no afecto en gran medida el funcionamiento del programa.

17- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se ha verificado que gran parte del problema en el reconocimiento es el hecho de que hay grandes cadenas de ceros, y por que las letras, aunque sean las mismas se escriben a diferentes tamaños, se implementa el zoom.Se usa la TDC, se regulariza lo segmentado.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Manuscrita	17	5	29.4%	3	3	100.0%	rho=0.3, n=30% de $tx*$ty
Manuscrita	17	4	23.5%	3	3	100.0%	rho=0.5, n=30% de $tx*$ty
Manuscrita	17	2	11.8%	3	3	100.0%	rho=1.0, n=30% de $tx*$ty
Manuscrita	17	4	23.5%	3	3	100.0%	rho=0.3, n=60% de $tx*$ty
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.5, n=60% de $tx*$ty
Manuscrita	17	2	11.8%	3	3	100.0%	rho=1.0, n=60% de $tx*$ty

Tabla A.3.17. Leve mejora en el reconocimiento, se ven cambios drásticos en los valores de los coeficientes de la TDC con pequeñas variaciones en las letras.

18- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se aplica zoom, no se usa la TDC, se ingresan directamente los datos a la red neuronal, se regularizan los tamaños de los caracteres segmentados.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Manuscrita	17	6	35.3%	3	3	100.0%	rho=0.3
Manuscrita	17	5	29.4%	3	3	100.0%	rho=0.4
Manuscrita	17	4	23.5%	3	3	100.0%	rho=0.5
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.8
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.9
Manuscrita	17	2	11.8%	3	3	100.0%	rho=1.0

Tabla A-3.18. Leve mejora en el reconocimiento, se aprecian que el usar pixeles individuales hace inestable el reconocimiento.

19- Para este caso se ocupaba la segmentación manual en el entrenamiento, se realizaron ajustes manuales en los parámetros del segmentador durante el funcionamiento normal del programa. Se aplica zoom, no se usa la TDC, se ocupa la media de un área para sacar los datos que van a la red neuronal, se regularizan los tamaños de los caracteres segmentados.

	LETRAS			LINEAS			
Tipo de Letra:	**En el texto**	**Reconocidas correctamente**	**% de acierto**	**En el texto**	**Reconocidas correctamente**	**% de aciertos**	**OBSERVACIONES**
Manuscrita	17	12	70.6%	3	3	100.0%	rho=0.3
Manuscrita	17	10	58.8%	3	3	100.0%	rho=0.4
Manuscrita	17	6	35.3%	3	3	100.0%	rho=0.5
Manuscrita	17	4	23.5%	3	3	100.0%	rho=0.8
Manuscrita	17	3	17.6%	3	3	100.0%	rho=0.9
Manuscrita	17	3	17.6%	3	3	100.0%	rho=1.0

Tabla A.3.19. Mejora apreciable en el reconocimiento.

A.4 Código de los módulos del ROCM

A.4.1 *IGU*

```
# Interfaz Gráfica de Usuario para el proyecto de OCR manuscrito
# Revisado el: 21/2/04

# Programa ppal. de ROCM - M, desde aquí se llaman a todas las funciones
y módulos a ocupar en el
# programa

# Autor: Yo - M

# MÓDULOS DE PERL
#Módulo para manejar ventanas en Perl
use Tk;
#Módulo de manejo de texto con "deshacer" incluido
use Tk::TextUndo;
#Para desplegar "balones" de ayuda en los botones
use Tk::Balloon;
# Para desplegar cajas de diálogo
use Tk::DialogBox;
# Para encontrar el directorio de trabajo automáticamente
use Cwd;
use Win32;
use Tk::Menubutton;     # Por las limitaciones de perl2exe
use Tk::Photo;
use Tk::Pane;           # Para el frame con scroll

# MODULOS PROPIOS
# Módulo del segmentador de imágenes (MI módulo), recordar que existen
dos parámetros de interés
# $segmentar::letras y $segmentar::lineas, donde se encuentra el número
de letras y el número de
# líneas encontradas
use segmentar;
#Módulo de la ayuda del segmentador (se necesitan 7 archivos de texto
(.ayu))
use ayuda;
# Módulo de entrenamiento para nuevos usuarios
use nusuario;
# Módulo que realiza el vecino8
use vecino8;
# Módulo de la red ART 2
use art2;
# Módulo de la identificación
use identificador;
# Módulo de corrección de lo identificado
use reglas;
# Módulo para variar el tamaño de las letras provenientes de la
segmentación
use zoom;

# Se borra la pantalla de la línea de comando
```

```
system("cls");
#Inicialización de algunas variables a ocupar durante el programa
#Variables para el segmentador
$tx=40;                 # Ancho de la ventana de segmentación
$ty=50;                 # Alto de la ventana de segmentación
$tolx=3;          # Espacio máximo en blanco que puede separar
$toly=3;          # dos puntos para decir si estan conectados
$usuario="ninguno";     # usuario inicial
$dir=cwd;         # asigno el nombre del directorio de trabajo a la var.
$dir
$entreno='no';          # Le digo al programa si se encuentra entrenando o
en su modo normal de
                  # trabajo, esta var. se modificará por nusuario y será
consultada por
                  #  segmentar,  y  posteriormente  por  art2,  para  tomar
condiciones especiales
                  # para el entrenamiento

#El nombre de la ventana principal es $ventanota
$ventanota = new MainWindow;
$ventanota->configure(-title=>"Interfaz Gráfica de Usuario ver 1.m");

#Pone las barras de menus
#$frmBarrademenuAyuda=$ventanota->Frame()->pack(-anchor=>'ne');
$frmBarrademenu=$ventanota->Frame()->pack(-anchor=>'nw');
$frmBarradeiconos=$ventanota->Frame()->pack(-anchor=>'nw');

# Botones del menú
$mbArchivo=$frmBarrademenu->Menubutton(-text=>'Archivo',-
activebackground=>"#cccccc",
          -relief=>'flat',-borderwidth=>1)->pack(-side=>'left');
$mbArchivo->command(-label=>'Nuevo',-accelerator=>'Alt+n',
                  -underline=>0,-command=>\&NuevoArchivo);
$mbArchivo->command(-label=>'Abrir       archivo       de       texto',-
accelerator=>'Alt+a',
                  -underline=>0,-command=>\&AbrirArchivo);
$mbArchivo->command(-label=>'Guardar archivo',-accelerator=>'Alt+g',
                  -underline=>0,-command=>\&GuardarArchivo);
$mbArchivo->command(-label=>'Guardar Archivo Como',
                        -command=>\&GuardarArchivoComo);
$mbArchivo->separator();
$mbArchivo->command(-label=>'Crea Nuevo Usuario',-accelerator=>'Alt+c',
                  -underline=>0,-command=>\&nusuario);
$mbArchivo->command(-label=>'Usuario',-accelerator=>'Alt+u',
                  -underline=>0,-command=>\&usuario);
$mbArchivo->command(-label=>"Guardar        Parámetros        Usuario",-
command=>\&gusuario);
$mbArchivo->separator();
$mbArchivo->command(-label=>'Salir',-accelerator=>'Alt+F4',-
command=>\&salir);


#Menu de edición
$mbEdicion=$frmBarrademenu->Menubutton(-text=>'Edición',-
activebackground=>"#cccccc",
          -relief=>'flat',-borderwidth=>1)->pack(-side=>'left');
```

```
      $mbEdicion->command(-label=>'Seleccionar                    todo',-
command=>\&SelTodo);
      $mbEdicion->command(-label=>'Borrar todo',-accelerator=>'Alt+b',
                        -underline=>0,-command=>\&BorrarTodo);
      $mbEdicion->command(-label=>'Reemplazar',-accelerator=>'Alt+r',
                        -underline=>0,-command=>\&Reemplazar);

#Menú de Procesamiento de imágenes y posteriormente también de
reconocimiento de caracteres
$mbProcesar=$frmBarrademenu->Menubutton(-text=>'Procesamiento',-
activebackground=>"#cccccc",
      -relief=>'flat',-borderwidth=>1)->pack(-side=>'left');
      $mbProcesar->command(-label=>'Parámetros     del     segmentador',-
command=>\&ParSeg);
      $mbProcesar->command(-label=>'Imagen          a          procesar',-
accelerator=>'Alt+i',
                  -underline=>0,-command=>\&AbrirIm);
      $mbProcesar->separator();
      $mbProcesar->command(-label=>'Corrección              automática',-
command=>\&Corrige);
      $mbProcesar->separator();
      $mbProcesar->command(-label=>'Guardar    pesos    de    la    red',-
command=>\&GuardaPesos);

#Menú de ayuda

$mbAyuda=$frmBarrademenu->Menubutton(-text=>'Ayuda',-
activebackground=>"#cccccc",
      -relief=>'flat',-borderwidth=>1)->pack(-anchor=>'ne');
      $mbAyuda->command(-label=>'Acerca de...',-command=>\&acercade);
      $mbAyuda->separator();
      $mbAyuda->command(-label=>'Ayuda           del           programa',-
accelerator=>'Alt+y',
                  -underline=>1,-command=>sub{&ayudaP});

#Menú de herramientas (iconos)
$btnNuevo=$frmBarradeiconos->Button(-command=>\&NuevoArchivo)->grid(-
column=>0,-row=>0);
      $icono=$btnNuevo->Photo(-file=>$dir.'\iconos\icono1.bmp');
            $btnNuevo->configure(-image=>$icono);
$btnAbrir=$frmBarradeiconos->Button(-command=>\&AbrirArchivo)->grid(-
column=>1,-row=>0);
      $icono=$btnAbrir->Photo(-file=>$dir.'\iconos\icono2.bmp');
            $btnAbrir->configure(-image=>$icono);
$btnGuardar=$frmBarradeiconos->Button(-command=>\&GuardarArchivo)->grid(-
column=>3,-row=>0);
      $icono=$btnGuardar->Photo(-file=>$dir.'\iconos\icono3.bmp');
            $btnGuardar->configure(-image=>$icono);
$btnNUsuario=$frmBarradeiconos->Button(-command=>\&nusuario)->grid(-
column=>4,-row=>0);
      $icono=$btnNUsuario->Photo(-file=>$dir.'\iconos\icono6.bmp');
            $btnNUsuario->configure(-image=>$icono);
$btnUsuario=$frmBarradeiconos->Button(-command=>\&usuario)->grid(-
column=>5,-row=>0);
      $icono=$btnUsuario->Photo(-file=>$dir.'\iconos\icono7.bmp');
            $btnUsuario->configure(-image=>$icono);
# Solo es un espacio para los botones en la barra de herramientas
```

```
$frmEspacio=$frmBarradeiconos->Frame(-width=>20)->grid(-column=>6,-
row=>0);

$btnParSeg=$frmBarradeiconos->Button(-command=>\&ParSeg)->grid(-
column=>7,-row=>0);
      $icono=$btnParSeg->Photo(-file=>$dir.'\iconos\icono4.bmp');
            $btnParSeg->configure(-image=>$icono);
$btnImagen=$frmBarradeiconos->Button(-command=>\&AbrirIm)->grid(-
column=>8,-row=>0);
      $icono=$btnImagen->Photo(-file=>$dir.'\iconos\icono5.bmp');
            $btnImagen->configure(-image=>$icono);
$frmEspacio2=$frmBarradeiconos->Frame(-width=>20)->grid(-column=>9,-
row=>0);
# Botón para aplicar reglas a la conversión
$btnCorrige=$frmBarradeiconos->Button(-command=>\&Corrige)->grid(-
column=>10,-row=>0);
      $icono=$btnCorrige->Photo(-file=>$dir.'\iconos\icono8.bmp');
            $btnCorrige->configure(-image=>$icono);
$btnReemplazar=$frmBarradeiconos->Button(-command=>\&Reemplazar)->grid(-
column=>11,-row=>0);
      $icono=$btnReemplazar->Photo(-file=>$dir.'\iconos\icono10.bmp');
            $btnReemplazar->configure(-image=>$icono);
# Espacio
$frmEspacio=$frmBarradeiconos->Frame(-width=>20)->grid(-column=>12,-
row=>0);
$btnAyuda=$frmBarradeiconos->Button(-command=>\&ayudaP)->grid(-
column=>13,-row=>0);
      $icono=$btnAyuda->Photo(-file=>$dir.'\iconos\icono9.bmp');
            $btnAyuda->configure(-image=>$icono);
#Balones de los íconos
$b=$ventanota->Balloon();
$b->attach($btnNuevo,
      -balloonmsg => "Nuevo documento de texto");
$b->attach($btnAbrir,
      -balloonmsg=>"Abrir un archivo de texto");
$b->attach($btnGuardar,
      -balloonmsg=>"Guardar archivo de texto");
$b->attach($btnNUsuario,
      -balloonmsg=>"Nuevo Usuario");
$b->attach($btnUsuario,
      -balloonmsg=>"Usuario");
$b->attach($btnParSeg,
      -balloonmsg=>"Parámetros del segmentador");
$b->attach($btnImagen,
      -balloonmsg=>"Abrir Imagen a Procesar");
$b->attach($btnCorrige,
      -balloonmsg=>"Corrección automática");
$b->attach($btnAyuda,
      -balloonmsg=>"Ayuda del programa");
$b->attach($btnReemplazar,
      -balloonmsg=>"Buscar y reemplazar texto");

#Creación y configuración de la caja de texto principal
$txtPrincipal = $ventanota->Scrolled('TextUndo',
                        -exportselection => 'true',
                        -setgrid => 'true',-wrap=>'none',
```

```
                        -scrollbars=>'osoe');       # Crea las barras y la
ventana
                                                    #las barras aparecen solo si son
                                                    #necesarias
$txtPrincipal->pack(-expand => 'yes' , -fill => 'both');
#tie  (*TEXT,  'Tk::TextUndo',$txtPrincipal);          #  se  puede  usar  un
manejador de archivos
                                    #,en este caso TEXT, para sacar inf.
directamente
                                    #a la ventana de texto como si fuera
la pantalla
# Se establece la fuente de la caja de texto principal
$txtPrincipal->configure(-font=>'Courier 10');

# Creación y configuración de la caja de texto secundaria
# En esta caja se muestran todos los mensajes y advertencias del programa
$txtSecundaria=$ventanota->Scrolled('Text',-scrollbars=>'osoe',-
height=>3)
                        ->pack(-expand=>'yes',-fill=>'both');

#EVENTOS DE COMBINACION DE BOTONES (mayúsculas y minúsculas)
$txtPrincipal->eventAdd('<<abrirarchivo>>'=>'<Alt-a>');
$txtPrincipal->eventAdd('<<abrirarchivo>>'=>'<Alt-A>');
$txtPrincipal->eventAdd('<<borrartodo>>'=>'<Alt-b>');
$txtPrincipal->eventAdd('<<borrartodo>>'=>'<Alt-B>');
$txtPrincipal->eventAdd('<<nuevousuario>>'=>'<Alt-C>');
$txtPrincipal->eventAdd('<<nuevousuario>>'=>'<Alt-c>');
$txtPrincipal->eventAdd('<<guardararchivo>>'=>'<Alt-g>');
$txtPrincipal->eventAdd('<<guardararchivo>>'=>'<Alt-G>');
$txtPrincipal->eventAdd('<<abririmagen>>'=>'<Alt-i>');
$txtPrincipal->eventAdd('<<abririmagen>>'=>'<Alt-I>');
$txtPrincipal->eventAdd('<<nuevoarchivo>>'=>'<Alt-n>');
$txtPrincipal->eventAdd('<<nuevoarchivo>>'=>'<Alt-N>');
$txtPrincipal->eventAdd('<<reemplazar>>'=>'<Alt-r>');
$txtPrincipal->eventAdd('<<reemplazar>>'=>'<Alt-R>');
$txtPrincipal->eventAdd('<<usuario>>'=>'<Alt-u>');
$txtPrincipal->eventAdd('<<usuario>>'=>'<Alt-U>');
$txtPrincipal->eventAdd('<<ayuda>>'=>'<Alt-y>');
$txtPrincipal->eventAdd('<<ayuda>>'=>'<Alt-Y>');
$txtPrincipal-
>bind('Tk::TextUndo','<<abrirarchivo>>',sub{&AbrirArchivo});
$txtPrincipal->bind('Tk::TextUndo','<<borrartodo>>',sub{&BorrarTodo});
$txtPrincipal->bind('Tk::TextUndo','<<nuevousuario>>',sub{&nusuario});
$txtPrincipal-
>bind('Tk::TextUndo','<<guardararchivo>>',sub{&GuardarArchivo});
$txtPrincipal->bind('Tk::TextUndo','<<abririmagen>>',sub{&AbrirIm});
$txtPrincipal-
>bind('Tk::TextUndo','<<nuevoarchivo>>',sub{&NuevoArchivo});
$txtPrincipal->bind('Tk::TextUndo','<<reemplazar>>',sub{&Reemplazar});
$txtPrincipal->bind('Tk::TextUndo','<<usuario>>',sub{&usuario});
$txtPrincipal->bind('Tk::TextUndo','<<ayuda>>',sub{&ayudaP});
# Condiciones Iniciales (nombre del archivo y foco)
$direcciondelarchivo="Nuevo.txt";
#establezco el foco en la caja de texto
$txtPrincipal->focus;
# Título de la ventana de texto principal
```

```
$ventanota->configure(-title=>"$titulo - Ubicación: $direcciondelarchivo.
Usuario: $usuario");
# Pongo el mensaje de listo en txtSecundario
$txtSecundaria->insert('end',"Programa listo\n");
MainLoop;
#_____________________________________________________________________
_____________________
#SUBRUTINAS


#SUBRUTINAS PARA EL MENU ARCHIVO
sub AbrirArchivo
{
      my $temporal;
#Se presenta el cuadro de diálogo para abrir un archivo
      $tipos=[['Texto','.txt'],
            ['Todos los archivos','*']];
      $temporal=$ventanota->getOpenFile(-filetypes=>$tipos);
# Se abre un archivo con el nombre especificado
# y se imprime en la caja de texto todo lo leído
# al final se cierra el archivo leído.
      if ($temporal ne "")
      {
            $direcciondelarchivo=$temporal;
            $txtPrincipal->Load($direcciondelarchivo);
            $ventanota->configure(-title=>"$titulo      -      Ubicación:
$direcciondelarchivo");
            $txtSecundaria->insert('end',"Abriendo
$direcciondelarchivo\n");
            $txtSecundaria->see('end');
      }
} #Fin de abrirarchivo

sub GuardarArchivo
{
my $temporal;

      if ($direcciondelarchivo eq "Nuevo.txt")
      {
            #Se presenta el cuadro de diálogo para abrir un archivo
            $tipos=[['Texto','.txt'],
                  ['Todos los archivos','*']];
            $temporal=$ventanota->getSaveFile(-filetypes=>$tipos,
                                    -defaultextension=>'.txt');
      }

# Si el nombre no es igual a "" entonces graba el contenido de la ventana
de texto principal en
# el archivo con el nombre especificado
      if ($temporal ne "")
      {
            $direcciondelarchivo=$temporal;
            $txtPrincipal->Save($direcciondelarchivo);
$ventanota->configure(-title=>"$titulo - Ubicación: $direcciondelarchivo.
Usuario: $usuario");
```

```
            $txtSecundaria->insert('end',"Guardando
$direcciondelarchivo\n");
            $txtSecundaria->see('end');
      }

} #Fin de GuardarArchivo

sub GuardarArchivoComo
{
      $direcciondelarchivo="Nuevo.txt";
      &GuardarArchivo;
} #Fin de GuardarArchivoComo

sub NuevoArchivo
{
      &BorrarTodo;
      $direcciondelarchivo="Nuevo.txt";
      $ventanota->configure(-title=>"$titulo          -          Ubicación:
$direcciondelarchivo. Usuario: $usuario");
} #Fin de NuevoArchivo

sub nusuario
{ #Llama a la rutina para crear un nuevo usuario del programa
      # Llama a la función para presentar los pasos
      $txtSecundaria->insert('end',"Creando nuevo usuario\n");
      $txtSecundaria->see('end');
      &nusuario::inicia;
      # Cuando termina el entrenamiento la rutina nusuario debe cambiar
el valor de
      # la variable $entreno
} #Fin de nusuario

sub usuario
{
#Llama a la rutina para cargar el archivo del usuario para personalizar
el programa
# Dicho archivo contendrá los pesos de la red y los parámetros del
segmentador que usa
      print "usuario\n";
      $dlgUsuario=$ventanota->DialogBox(-title=>'Nombre del usuario',
                  -buttons=>['Aceptar','Cancelar']);
      $entUsuario=$dlgUsuario->add('Entry')->pack;
      $entUsuario->focus;
# Muestra el cuadro de diálogo
      $btnDialogo=$dlgUsuario->Show();
# Se evalúa el botón presionado
      if ($btnDialogo eq 'Aceptar')
      {
            $usuario=$entUsuario->get();
            if ($usuario ne "")
            {
                  $SegUsuario=$dir."\\usuarios\\".$usuario.".par";
                  if(open(PARAMETROS,"$SegUsuario"))
                  {
                        chomp($tx=<PARAMETROS>);
                        chomp($ty=<PARAMETROS>);
                        chomp($tolx=<PARAMETROS>);
```

```
                        chomp($toly=<PARAMETROS>);
                        chomp($nEnt=<PARAMETROS>);
                        chomp($txini=<PARAMETROS>);
                        chomp($tyini=<PARAMETROS>);
$ventanota->configure(-title=>"$titulo - Ubicación: $direccciondelarchivo.
Usuario: $usuario");
                        $txtSecundaria->insert('end',"¡Bienvenido
$usuario!\n");
                        $txtSecundaria->see('end');
                        $txtSecundaria->insert('end',"Parametros       del
segmentador modificados a Tx $tx, Ty $ty, Tolerancia x $tolx Tolerancia y
$toly\n");
                        $txtSecundaria->see('end');
                  }
                  else
                  {
                        $txtSecundaria->insert('end',"¡El    usuario    no
existe!\n");
                        $txtSecundaria->see('end');
                        $ventanota->bell;
                  }
            }

            print "presionado Aceptar $SegUsuario\n";

      }
      else
      {
            print "cancelado\n";
      }


} #Fin de usuario

sub gusuario
{
      if (PARAMETROS!="")
      {     #Se encarga de guardar los cambios hechos en el usuario
escogido
            print PARAMETROS "$tx\n";
            print PARAMETROS "$ty\n";
            print PARAMETROS "$tolx\n";
            print PARAMETROS "$toly\n";
            print PARAMETROS "$nEnt\n";
            print PARAMETROS "$txini\n";
            print PARAMETROS "$tyini\n";
            $txtSecundaria->insert('end',"¡Modificado $usuario!\n");
            $txtSecundaria->see('end');
      }
      else
      {
            $txtSecundaria->insert('end',"¡No    se    ha    escogido    un
usuario!\n");
            $txtSecundaria->see('end');
            $ventanota->bell;
      }
```

```
}

sub salir
{ #Termina el programa

      $ventanota->destroy;

} #Fin de salir
#*****************************
#SUBRUTINAS PARA EL MENÚ EDICIÓN

sub SelTodo
{ #Selecciona todo el texto que aparece en la pantalla

#     print "selecciona todo\n";
      $txtPrincipal->tagAdd('sel','0.0','end');
      $txtSecundaria->insert('end',"Se ha seleccionado todo el texto\n");
      $txtSecundaria->see('end');

} # Fin de SelTodo

sub BorrarTodo
{
      $txtPrincipal->delete('1.0','end');
      $txtSecundaria->insert('end',"Se ha borrado todo el texto\n");
      $txtSecundaria->see('end');
} #Fin de BorrarTodo

sub Reemplazar
{
# Presenta una caja de diálogo para el reemplazo básico, no expresiones
regulares
      print "Reemplazar\n";
# La variable encuentra puede ser modificada para empezar desde otro
punto para no caer en la
# misma coincidencia
      $encuentra="0.0";
      $vntReemplazar=$ventanota->Toplevel();
      $vntReemplazar->configure(-title=>'Reemplazo de palabras');
      $vntReemplazar->focus;
      $lblInf=$vntReemplazar->Label(-text=>'Ingrese la palabra que desea
reemplazar')
      ->grid(-column=>0,-row=>0);
      $entBuscar=$vntReemplazar->Entry()->grid(-column=>0,-row=>1,-
sticky=>'ew');
      $entCon=$vntReemplazar->Entry()->grid(-column=>0,-row=>3,-
sticky=>'ew');
      $btnBuscar=$vntReemplazar->Button(-text=>'Buscar        Siguiente',-
command=>\&Buscar)
      ->grid(-column=>1,-row=>0,-sticky=>'ew');
      $lblInf=$vntReemplazar->Label(-text=>'Ingrese la palabra con que
desea reemplazar lo encontrado')
      ->grid(-column=>0,-row=>2);
      $btnR=$vntReemplazar->Button(-text=>'Reemplazar',-command=>\&Reemp)
      ->grid(-column=>1,-row=>1,-sticky=>'ew');
      $btnRT=$vntReemplazar->Button(-text=>'Reemplazar           todas',-
command=>\&ReempT)
```

```
      ->grid(-column=>1,-row=>2,-sticky=>'ew');
      $btnCancelar=$vntReemplazar->Button(-text=>'Cancelar',-
command=>sub{
      $vntReemplazar->destroy;
      $txtPrincipal->focus;})->grid(-column=>1,-row=>3,-sticky=>'ew');
      $entBuscar->focus;
} # Fin de Reemplazar

sub Buscar
{
# Se ha presionado el botón para Buscar siguiente
# Se obtiene el texto a buscar
      my $TextoBuscar=$entBuscar->get;
# Se obtiene el texto con el que se desea reemplazar lo encontrado
      my $TextoCon=$entCon->get;
# Encuentro el número de caracteres que posee el texto buscar
      my @temp=split(//,$TextoBuscar);
      my $nTB=@temp;
      $nTB="$nTB"."chars";
      print "posee $nTB caracteres\n";

      $encuentra=$txtPrincipal->search(-
regexp,"$TextoBuscar","$encuentra");
#print "encuentra $encuentra";
      if($encuentra ne "")
      {
      # Muestra una caja de texto para preguntar si se desea o no
reemplazar
            $txtPrincipal->see($encuentra);
            $txtPrincipal->insert("$encuentra","");
            $txtPrincipal->focus;
            $vntReemplazar->focus;
            $dlgReemp=$vntReemplazar->DialogBox(-title=>'Se ha encontrado
una coincidencia',
                              -buttons=>['Aceptar','Cancelar']);
            $lblUsuario=$dlgReemp->add('Label')->pack;
            $lblUsuario->configure(-text=>"Se    encontro    $TextoBuscar
¿desea reemplazarlo con $TextoCon?");
      # Muestra el cuadro de diálogo
            $btnDialogo=$dlgReemp->Show();
            if($btnDialogo eq 'Aceptar')
            {
            # Si acepta modificar la palabra se selecciona el texto lo
borra y pone la nueva palabra
                  $txtPrincipal->delete("$encuentra","$encuentra+$nTB");
                  $txtPrincipal->insert("$encuentra","$TextoCon");
            }
      # Modifico encuentra
            $encuentra="$encuentra+1chars";
      }
      else
      {
      # Muestra el mensaje de no se ha encontrado el texto
            print "No existe $TextoBuscar\n";
            $dlgReemp=$vntReemplazar->DialogBox(-title=>'No  encontró  una
coincidencia',
                        -buttons=>['Aceptar']);
```

```
            $lblUsuario=$dlgReemp->add('Label')->pack;
            $lblUsuario->configure(-text=>"No     se     ha     encontrado
$TextoBuscar en el texto.");
      # Muestra el cuadro de diálogo
            $btnDialogo=$dlgReemp->Show();
      # Pongo esto para evitar algún error pues el indice queda en ""
      $encuentra='0.0';
      }

} # Fin de Buscar

sub Reemp
{
# Se reemplaza lo encontrado con lo puesto en la entrada entCon
# Se obtiene el texto a buscar
      my $TextoBuscar=$entBuscar->get;
# Se obtiene el texto con el que se desea reemplazar lo encontrado
      my $TextoCon=$entCon->get;
# Encuentro el número de caracteres que posee el texto buscar
      my @temp=split(//,$TextoBuscar);
      my $nTB=@temp;
      $nTB="$nTB"."chars";
      $encuentra=$txtPrincipal->search(-
regexp,"$TextoBuscar","$encuentra");
#print "encuentra $encuentra";
      if($encuentra ne "")
      {
      # Muestra una caja de texto para preguntar si se desea o no
reemplazar
            $txtPrincipal->see($encuentra);
            $txtPrincipal->insert("$encuentra","");
            $txtPrincipal->focus;
            $vntReemplazar->focus;
      # Si acepta modificar la palabra se selecciona el texto lo borra y
pone la nueva palabra
            $txtPrincipal->delete("$encuentra","$encuentra+$nTB");
            $txtPrincipal->insert("$encuentra","$TextoCon");
      # Modifico encuentra
            $encuentra="$encuentra+1chars";
      }
      else
      {
      # Muestra el mensaje de no se ha encontrado el texto
            print "No existe $TextoBuscar\n";
            $dlgReemp=$vntReemplazar->DialogBox(-title=>'No     se     ha
encontrado una coincidencia',
                  -buttons=>['Aceptar']);
            $lblUsuario=$dlgReemp->add('Label')->pack;
            $lblUsuario->configure(-text=>"No     se     ha     encontrado
$TextoBuscar en el texto.");
      # Muestra el cuadro de diálogo
            $btnDialogo=$dlgReemp->Show();
      # Pongo esto para evitar algún error pues el indice queda en ""
      $encuentra='0.0';
      }
} # Fin De reempt
```

```
#*****************************
#SUBRUTINAS PARA EL MENU PROCESAMIENTO

sub ParSeg
{ #Llama a la ventana para establecer los parámetros del segmentador
# Aunque ya tendrá parámetros por defecto que serán especificados al
inicio de este programa

      print "Parámetros del segmentador\n";
      $popParSeg=$frmBarrademenu->Toplevel();
      $popParSeg->title("Parámetros del segmentador\n");
      $popParSeg->Label(-text=>"Introduzca    los    valores    para    el
segmentador: ")->grid(-column=>0,-row=>0,-columnspan=>2);
      $lblTx=$popParSeg->Label(-text=>"Tx")->grid(-column=>0,-row=>1);
      $entTx=$popParSeg->Entry()->grid(-column=>1,-row=>1);
      $lblTy=$popParSeg->Label(-text=>"Ty")->grid(-column=>0,-row=>2);
      $entTy=$popParSeg->Entry()->grid(-column=>1,-row=>2);
      $lblTolx=$popParSeg->Label(-text=>"Tol        x")->grid(-column=>0,-
row=>3);
      $entTolx=$popParSeg->Entry()->grid(-column=>1,-row=>3);
      $lblToly=$popParSeg->Label(-text=>"Tol        y")->grid(-column=>0,-
row=>4);
      $entToly=$popParSeg->Entry()->grid(-column=>1,-row=>4);
      print "$tx $ty $tolx $toly\n";
      $entTx->insert("0.0","$tx");
      $entTy->insert("0.0","$ty");
      $entTolx->insert("0.0","$tolx");
      $entToly->insert("0.0","$toly");
      $btnAceptar=$popParSeg->Button(-text=>"Aceptar",-
command=>\&rutParSeg)->grid(-column=>0,-row=>5);
      $btnCancelar=$popParSeg->Button(-text=>"Cancelar",-
command=>sub{$popParSeg->destroy();
                        $txtPrincipal->focus})->grid(-column=>1,-
row=>5,
                                                -sticky=>'e',-padx=>8);
# Pone el enfoque en la caja
      $popParSeg->focus;
      $popParSeg->resizable('no','no');
} #Fin de ParSeg

sub AbrirIm
{ #Llama al cuadro de abrir archivo para obtener la imagen a procesar,
después deberá llamar a
# Mostrar imagen de segmentar para mostrarla en otra ventana
      $tipos=[['Imagen',['.bmp','.gif']]];
      $dirIm=$ventanota->getOpenFile(-filetypes=>$tipos);
#Se abre un archivo con el nombre especificado

      if ($dirIm ne "")
      {
            #Debo hacer una sustitución de / por \ para que pueda llamar
            #al archivo correcto
            $txtSecundaria->insert('end',"Se    trabajará    el    archivo
$dirIm\n");
            $txtSecundaria->see('end');
            $dirImCorta=Win32::GetShortPathName($dirIm);
```

```
            print "dir de Imagen $dirImCorta\n";
            &segmentar::MostrarI($dirImCorta);
      }
}

sub rutParSeg
{ #Rutina para obtener los parámetros de popParSeg
      $tx=$entTx->get;  #Obtiene los datos de Tx
      $ty=$entTy->get;  # datos de ty
      $tolx=$entTolx->get;    # datos de tolx
      $toly=$entToly->get;    # datos de toly
      print "Se tiene tx $tx ty $ty tolx $tolx toly $toly\n";
      $popParSeg->destroy;    # Destruye la ventana de donde fue llamada
      $txtSecundaria->insert('end',"Parametros       del       segmentador
modificados a Tx $tx, Ty $ty, Tolx $tolx toly $toly\n");
      $txtSecundaria->see('end');
}

sub Corrige
{
# Llama a la función corrige del módulo reglas, dicha función se encarga
de actualizar el
# contenido de la vetana de texto principal

      &reglas::corrige($hilera);
}

sub GuardaPesos
{
# Guarda los pesos del usuario
$PesUsuario=$dir."\\usuarios\\".$usuario.".pes";
&art2::art2_guardar($PesUsuario);
}

#*****************************
#SUBRUTINAS PARA EL MENÚ DE AYUDA
sub acercade
{
      $pop = $frmBarrademenu->Toplevel();
      $pop->title("Acerca de");
      $pop->Label(-text=>"ROCM    Reconocedor    Óptico    de    Caracteres
Manuscritos (M)")->pack();
      $pop->Label(-text=>"Ver. 1.0")->pack();
      $pop->Label(-text=>"Todos los derechos reservados")->pack();
      $pop->Label(-text=>"Miguel E. Flores G.")->pack();
      $pop->Label(-text=>"Sin sitio en Internet")->pack();

# Le paso el foco a la inf acerca de
      $pop->focus;
            my $button_ok = $pop->Button(text=>'OK',
                  command => sub {$pop->destroy();
                  $txtPrincipal->focus;
                  } )
                  ->pack();
            $pop->resizable('no','no');

} #Fin de acercade
```

```
sub ayudaP
{ #Llama a la función principal del módulo de ayuda
      $txtSecundaria->insert('end',"Consultando      la      ayuda      del
programa\n");
      $txtSecundaria->see('end');
      &ayuda::MostrarAyuda;
      #&ayuda::Tema(2);
      # Le paso el enfoque a la ventana de ayuda
      $ayuda::vayuda->focus;

}
```

A.4.2 *Ayuda*

```
# Ayuda para el proyecto ROCM - M
# Los archivos con el texto de ayuda deben encontrarse en el mismo
directorio que éste programa
# tendrán extensión .ayu y serán enumerados del 1 en adelante
# Revisado 27/2/04
# Autor: Yo - M

# Funciones:
# MostrarAyuda          Muestra la página de inicio de la ayuda
# MostrarInicio         Crea el índice de la ayuda
# Tema($navegante)      Llama al tema correspondiente a $navegante
# InIm($figura,$pos)    Inserta la imagen en la ventana de ayuda en la pos
pasada como argumento
#                 en el formato 'fila.columna', fig es un número (1-?)
package ayuda;
#_____________________________________________________________________________
_____________________
#     SUBRUTINAS

sub MostrarAyuda
{ #Se encarga de mostrar la ventana de ayuda y la página de inicio
# Se pondrá la página de inicio de ayuda, de allí pasará a las otras
cuando sea necesario
# Aquí se debería poner:
      $vayuda=$main::ventanota->Toplevel();
      $vayuda->configure(-title=>'Ayuda de ROCM - M');
      # La variable $navegante se encarga de saber y notificar la página
en que se encuentra
      # o dirige el usuario
      $navegante=0;     # El valor 0 equivale a la págína de inicio
      $lblTitulo=$vayuda->Label(-text=>"Ayuda  para  ROCM",-font=>"tahoma
20 bold")
                              ->pack(-anchor=>'nw');
      $frmAyuda=$vayuda->Frame()->pack(-anchor=>'nw');
      $btnInicio=$frmAyuda->Button(-text=>"INICIO",-
command=>sub{&MostrarInicio;})
                              ->grid(-column=>0,-row=>0);
      $btnAnterior=$frmAyuda->Button(-text=>"<<                ANTERIOR",-
command=>sub{&Tema(--$navegante)})
```

```
                              ->grid(-column=>1,-row=>0);
      $btnSiguiente=$frmAyuda->Button(-text=>"SIGUIENTE                >>",-
command=>sub{&Tema(++$navegante)})
                              ->grid(-column=>2,-row=>0);
      $btnSalir=$frmAyuda->Button(-text=>"SALIR",-command=>sub{$vayuda-
>destroy;})
                        ->grid(-column=>3,-row=>0);
      $txtPrincipal=$vayuda->Scrolled('Text',-wrap=>'word',-
font=>'courier 8 bold',
                              -scrollbars=>'oe')
                              ->pack(-fill=>'both',-expand=>'yes');

      &MostrarInicio;         #Llama a la función para mostrar la página
de inicio
} #Fin de MostrarAyuda

#*******************************************
sub MostrarInicio
{ #Muestra la página de inicio (índice de la ayuda)
      $navegante=0;
      #Configuro el botón Anterior como deshabilitado pues nos
encontramos en la pagina de
      #Inicio
      $btnAnterior->configure(-state=>'disabled');
      $btnSiguiente->configure(-state=>'normal');
# Definición de los temas a presentar
      $tema1="1- ¿Qué es el reconocimiento óptico de caracteres?.";
      $tema2="2- Generalidades acerca del programa ROCM - M.";
      $tema3="3- Creación de nuevos usuarios y entrenamiento de la red.";
      $tema4="4- Usuarios ya creados (Como ingresar y realizar un
reconocimiento) y definición de los pesos de la red.";
      $tema5="5- Reconocimiento no satisfactorio.";
      $tema6="6- Límites del programa.";
      $tema7="7- Acerca de este programa.";
      $ee="\n\n   ";
# Definición de los estilos ocupados
      @resalta=(qw/-background black foreground white relief raised
underline 1/);
      @normal=(-background=>'white',-foreground=>'#ff00ff',-
relief=>'flat',-underline=>0);

      $txtPrincipal->configure(-state=>'normal');
      $txtPrincipal->delete('0.0','end');
      $txtPrincipal->insert('end'," Ayuda del programa Reconocedor Óptico
de Caracteres Manuscritos. Seleccione el tema sobre el que quiera
consultar de la siguiente lista:");
      $txtPrincipal->insert('end',"$ee");
      $txtPrincipal->insert('end',"$tema1",'t1');
      $txtPrincipal->insert('end',"$ee");
      $txtPrincipal->insert('end',"$tema2",'t2');
      $txtPrincipal->insert('end',"$ee");
      $txtPrincipal->insert('end',"$tema3",'t3');
      $txtPrincipal->insert('end',"$ee");
      $txtPrincipal->insert('end',"$tema4",'t4');
      $txtPrincipal->insert('end',"$ee");
      $txtPrincipal->insert('end',"$tema5",'t5');
      $txtPrincipal->insert('end',"$ee");
```

```
	$txtPrincipal->insert('end',"$tema6",'t6');
	$txtPrincipal->insert('end',"$ee");
	$txtPrincipal->insert('end',"$tema7",'t7');
	foreach $etiqueta (qw(t1 t2 t3 t4 t5 t6 t7))
	{
		$txtPrincipal->tagConfigure($etiqueta,@normal);
	}

# Activación y desactivación de las etiquetas al pasar el ratón sobre
ellas
	$txtPrincipal->tagBind("t1",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t1",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t1",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t1",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t2",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t2",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t2",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t2",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t3",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t3",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t3",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t3",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t4",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t4",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t4",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t4",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t5",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t5",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t5",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t5",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t6",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t6",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t6",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t6",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
	$txtPrincipal->tagBind("t7",'<Any-Enter>'=>
	sub{$txtPrincipal->tagConfigure("t7",@resalta);
	$txtPrincipal->configure(-cursor=>hand2);});
	$txtPrincipal->tagBind("t7",'<Any-Leave>'=>
	sub{$txtPrincipal->tagConfigure("t7",@normal);
	$txtPrincipal->configure(-cursor=>xterm);});
# Eventos asociados a las etiquetas al presionar el ratón encima de ellas
	$txtPrincipal->tagBind('t1','<1>',sub{&Tema(1)});
	$txtPrincipal->tagBind('t2','<1>',sub{&Tema(2)});
	$txtPrincipal->tagBind('t3','<1>',sub{&Tema(3)});
	$txtPrincipal->tagBind('t4','<1>',sub{&Tema(4)});
```

```
	$txtPrincipal->tagBind('t5','<1>',sub{&Tema(5)});
	$txtPrincipal->tagBind('t6','<1>',sub{&Tema(6)});
	$txtPrincipal->tagBind('t7','<1>',sub{&Tema(7)});

	#$txtPrincipal->configure(-state=>'disabled');


} #Fin de MostrarInicio

#*******************************************
sub Tema
{ #Llama al archivo indicado en su argumento
	$txtPrincipal->configure(-state=>'normal');
	$navegante=$_[0];
	if($navegante==0)
	{
	# Si navegante es cero entonces termina esta función y muestra el
inicio de la
	# ayuda
		&MostrarInicio;
		return();
	}
	# Obtengo el dir de trabajo y le agrego el dir ayuda
	$camino=$main::dir."\\ayuda\\";
	open(AYUDA,$camino."$navegante".".ayu")   or   die   ("No   se   pudo
encontrar el archivo");
# Se borra el texto para poner lo encontrado en el archivo
	$txtPrincipal->delete('0.0','end');
# Obtengo la primera línea del archivo, ese es el título
	$texto=<AYUDA>;
# Creo el formato titulo

	$txtPrincipal->insert('end',$texto,'titulo');
	$txtPrincipal->tagConfigure('titulo',-font=>'courier 14 bold');
	while (<AYUDA>)
	{
		$txtPrincipal->insert('end',$_);
	}

# Se habilitan o deshabilitan los botones ANTERIO Y SIGUIENTE
	if ($navegante>0)
	{
		$btnAnterior->configure(-state=>'normal');
	}
	else
	{
		$btnAnterior->configure(-state=>'disabled');
	}
	if ($navegante>6)
	{
		$btnSiguiente->configure(-state=>'disabled');
	}
	else
	{
		$btnSiguiente->configure(-state=>'normal');
	}
```

```
# Aquí se colocan los llamados a la rutina para insertar imágenes, se le
pasa el
# número de la fig y la pos (fila.columna) donde debe aparecer
my $fig=0;
my $pos='1.0';
      if ($navegante==1)
      {
      # Figuras correspodientes a la intro a ocr
      $fig='inicio';
      $pos='end';
      &InIm($fig,$pos);
      }
      elsif ($navegante==2)
      {
      $fig='describeigu';
      $pos='end';
      &InIm($fig,$pos);
      }
      elsif ($navegante==3)
      {
      $fig='paso1';
      $pos='12.0';
      &InIm($fig,$pos);
      $fig='paso2';
      $pos='18.0';
      &InIm($fig,$pos);
      $fig='paso3';
      $pos='24.0';
      &InIm($fig,$pos);
      $fig='paso4';
      $pos='30.0';
      &InIm($fig,$pos);
      $fig='procesar';
      $pos='34.0+5chars';
      &InIm($fig,$pos);
      $fig='RPaso1';
      $pos='40.0';
      &InIm($fig,$pos);
      }
      elsif ($navegante==4)
      {
      $fig='ingreseusuario';
      $pos='6.0';
      &InIm($fig,$pos);
      $fig='abririm';
      $pos='21.0';
      &InIm($fig,$pos);
      $fig='procesar';
      $pos='25.0';
      &InIm($fig,$pos);
      $fig='corrige';
      $pos='33.0';
      &InIm($fig,$pos);
      $fig='guardar';
      $pos='39.0';
      &InIm($fig,$pos);
      $fig='icoreemplazo';
```

```
        $pos='46.0';
        &InIm($fig,$pos);
        $fig='vntreemplazo';
        $pos='49.0';
        &InIm($fig,$pos);
        }
        elsif ($navegante==5)
        {

        }
        elsif ($navegante==6)
        {

        }
        else
        {
        # Cuando el paso es 7

        }
}
#*********************************************
sub InIm
{
# Rutina para insertar imágenes en la ayuda del programa
$txtPrincipal->tag(qw/configure centrar -justify center/);
        my $fig=$_[0].'.bmp';
        my $pos1=$_[1];
        my $pos2=$pos1.'+1chars';
        $lblIm1=$txtPrincipal->Label();
        $txtPrincipal->insert("$pos1","\n");
        $txtPrincipal->window('create',"$pos2",-window=>$lblIm1);
        $i=$camino.$fig;
        $imagen=$lblIm1->Photo(-file=>"$i");
        $lblIm1->configure(-image=>$imagen);
        $pos2=$pos1.'+2chars';
        $txtPrincipal->insert("$pos2","\n");
        print "Texto insertado en $pos2\n";
        #Una vez introducida la ayuda se deshabilita de nuevo la caja de
texto
        #$txtPrincipal->configure(-state=>'disabled');

}
#*********************************************
1;
```

A.4.3 *nusuario*

```
# Presenta el diálogo para la creación de nuevos usuarios y al mismo
entrena la red
# para ajustarse a la escritura del mismo y mejorar así el reconocimiento
# Revisado: 26/2/04

# Se encarga de llamar al segmentador, a la tdc y de entrenar a la red
neuronal
# Falta arreglar la verificación, aunque se podría obviar (paso 5)
# Arreglado para funcionar con:
```

```
# zoom, art2, reglas, vecino8.
package nusuario;

#_____________________________________________________________________
_____________________
#     SUBRUTINAS


sub inicia
{
system("cls");
#Esta rutina se encarga de dirigir todo el proceso
# En esta matriz se guarda la relación entre las letras y la salida de la
red que las activa
# La pos. en la matriz es la salida activada y el contendio es la letra
asociada
      @relacion=();
# Matriz con los caracteres que reconocerá
@caracteres=qw|A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z a b
c d e f g h i j k l m n ñ o p q r s t u v w x y z á é í ó ú ü 0 1 2 3 4 5
6 7 8 9 . ¡ ! ¿ ? + - * / , " $ < > [ ] ( ) { } ;|;
# Posición en caracteres, caracter al que se esta apuntando en la matriz
caracteres
      $posCar=0;
      $paso=0;
      $main::btnNUsuario->configure(-state=>'disabled');
# La variable $paso se encarga de llevar en ella el número del paso
actual para elegir la rutina
# apropiada en la función procesar
      $vntDialogo=$main::ventanota->Toplevel();
      $camino=$main::dir."\\pasos\\";
      $vntDialogo->configure(-title=>'Creación de nuevo usuario y
personalización de la red');
# Indico que esta en proceso un entrenamiento
      $main::entreno='si';
      $main::txtSecundaria->insert('end',"Valor de entreno en nusuario
$main::entreno\n");
# Etiquetas
      $lblDibujos=$vntDialogo->Label()->pack(-anchor=>'n');
      $imagen=$lblDibujos->Photo(-file=>$camino."paso0.bmp");
      $lblDibujos->configure(-image=>$imagen);
# Frame y botones
      $frmBotones=$vntDialogo->Frame()->pack(-anchor=>'n',-fill=>'x',-
expand=>'yes');
      $btnCancelar=$frmBotones->Button(-text=>'Cancelar',-command=>sub{
                  $main::entreno='no';$vntDialogo->destroy;
                  $main::btnNUsuario->configure(-state=>'normal');
                  $main::txtSecundaria->insert('end',"Entreno
$main::entreno\n");
                  $main::txtSecundaria->see('end')})
                  ->grid(-column=>1,-row=>0,-padx=>80);
      $btnSiguiente=$frmBotones->Button(-text=>'Siguiente',-
command=>sub{&pasoA;})
                  ->grid(-column=>2,-row=>0);
      $btnSiguiente->focus;
      $vntDialogo->focus;
# Cuando se destruye hace algo
```

```
      $vntDialogo->protocol('WM_DELETE_WINDOW'=>sub{
                  $main::txtSecundaria->insert('end','Ventana de nusuario
destruida');
                  $main::txtSecundaria->see('end');
                  $vntDialogo->destroy;$main::entreno='no';
                  $main::txtSecundaria->see('end');
                  $main::btnNUsuario->configure(-state=>'normal');}
                  );
} #Fin de Inicio

#***********************************

sub pasoA
{
#Esta rutina se encarga de manejar los pasos del 1 al 4, solo cambia el
texto y la imagen que
#se muestra en cada paso
# Primer paso del entrenamiento
      $vntDialogo->destroy();
      $vntDialogo=$main::ventanota->Toplevel();
      $vntDialogo->configure(-title=>"PASO 1");
      $paso=1;    # Indica procesar para el paso 1
      $lblPaso=$vntDialogo->Label(-text=>"PASO 1: Se debe ingresar una
imagen donde se muestren los siguientes caracteres tal y como aparecen en
la figura:",
            -font=>'courier      14      bold',-wraplength=>'15c')->pack(-
anchor=>'n');
      $lblPasoF=$vntDialogo->Label()->pack(-anchor=>'n');
      $imagen=$lblPasoF->Photo(-file=>$camino."paso1.bmp");
      $lblPasoF->configure(-image=>$imagen);
# Frame y botones
      $frmBotones=$vntDialogo->Frame()->pack(-anchor=>'n',-fill=>'x',-
expand=>'yes');
      $btnAbrirIm=$frmBotones->Button(-text=>'Abrir              Imagen',-
command=>sub{&main::AbrirIm})
                  ->grid(-column=>0,-row=>0);
      $btnAbrirIm->focus;
      $btnCancelar=$frmBotones->Button(-text=>'Cancelar',-
command=>sub{$main::entreno='no';
                  $main::btnNUsuario->configure(-state=>'normal');
                  $main::txtSecundaria->insert('end',"entreno
$main::entreno\n");
                  $main::txtSecundaria->see('end');
                  $vntDialogo->destroy;})
                  ->grid(-column=>1,-row=>0,-padx=>80);
      $btnSiguiente=$frmBotones->Button(-text=>'Siguiente',-
state=>'disabled',
                              -command=>sub{
            if ($paso==1)
            {
                  $nEnt=int($tx*$ty*1.0);
            # Se guardan los valores iniciales de tx y ty
                  $txini=$tx;
                  $tyini=$ty;
                  # Configuración de la red ART
                  print "Se ha configurado la red desde nusuario con
$nEnt entradas\n";
```

```
                &art2::art2Inicio($nEnt);
                #Cuando es el paso 1 y se presiona sig. se activa el
paso 2
                $paso=2;
                $btnSiguiente->configure(-state=>'disabled');
                $vntDialogo->configure(-title=>"PASO 2");
                $lblPaso->configure(-text=>'PASO 2: Se debe ingresar
una imagen donde se muestren los siguientes caracteres tal y como
aparecen en la figura:');
                $imagen=$lblPasoF->Photo(-file=>$camino."paso2.bmp");
                $lblPasoF->configure(-image=>$imagen);
                $btnAbrirIm->focus;
          }
          elsif ($paso==2)
          {

                #Cuando es el paso 2 y se presiona sig. se activa el
paso 3
                $paso=3;
                $btnSiguiente->configure(-state=>'disabled');
                $vntDialogo->configure(-title=>"PASO 3");
                $lblPaso->configure(-text=>'PASO 3: Se debe ingresar
una imagen donde se muestren los siguientes caracteres tal y como
aparecen en la figura:');
                $imagen=$lblPasoF->Photo(-file=>$camino."paso3.bmp");
                $lblPasoF->configure(-image=>$imagen);
                $btnAbrirIm->focus;
          }
          elsif ($paso==3)
          {
                #Cuando es el paso 3 y se presiona sig. se activa el
paso 4
                $paso=4;
                $btnSiguiente->configure(-state=>'disabled');
                $vntDialogo->configure(-title=>"PASO 4");
                $lblPaso->configure(-text=>'PASO 4: Se debe ingresar
una imagen donde se muestren los siguientes caracteres tal y como
aparecen en la figura:');
                $imagen=$lblPasoF->Photo(-file=>$camino."paso4.bmp");
                $lblPasoF->configure(-image=>$imagen);
                $btnAbrirIm->focus;
          }
          elsif ($paso==4)
          {
                $btnAbrirIm->destroy();
                # Se cambia a no entreno en la art2
                $art2::entreno='no';
                $paso=5;
                $btnSiguiente->configure(-state=>'normal',-
text=>'Finalizar');
                $vntDialogo->configure(-title=>"PASO 5");
                $lblPasoF->destroy();
                $lblPaso->configure(-text=>'PASO 5: Presione finalizar
para terminar la creación del nuevo usuario, aparecerá un cuadro de
diálogo para ingresar el nombre del usuario creado. Presione Cancelar
para terminar el proceso sin crear ningún usuario.');
```

```
                    # Establezco $main::entreno a "no" para que corra la
función segmentar
                    # normalmente, al final el usuario decidirá si el texto
esta o no
                    # correcto pudiendo elegir entre realizar el entreno de
nuevo o finalizar
                    # guardando los parámetros obtenidos
                    $main::entreno="no";
              }
              else
              {
                    # Cuando se presiona el botón siguiente (finalizar) en
el paso 5 se crea
                    # el archivo donde se guardarán los parámetros del
usuario.
NOMBREMALO:
      $dlgUsuario=$vntDialogo->DialogBox(-title=>'Nombre del usuario',
                        -buttons=>['Aceptar','Cancelar']);
      $entUsuario=$dlgUsuario->add('Entry')->pack;
      $entUsuario->focus;
# Muestra el cuadro de diálogo
      $btnDialogo=$dlgUsuario->Show();
# Se evalúa el botón presionado
      if ($btnDialogo eq 'Aceptar')
      {
            $usuario=$entUsuario->get();
            if ($usuario ne "")
            {
                  $SegUsuario=$main::dir."\\usuarios\\".$usuario.".par";
                  if(open(PARAMETROS,">$SegUsuario"))
                  {
                        print PARAMETROS "$tx\n";
                        print PARAMETROS "$ty\n";
                        print PARAMETROS "$tolx\n";
                        print PARAMETROS "$toly\n";
                        print PARAMETROS "$nEnt\n";
                        print PARAMETROS "$txini\n";
                        print PARAMETROS "$tyini\n";
                        $main::txtSecundaria->insert('end',"¡Creado
$usuario!\n");
                        $main::txtSecundaria->see('end');
                        $vntDialogo->destroy;
                        close(PARAMETROS);
                        $main::btnNUsuario->configure(-state=>'normal');
                        my $tRel=$main::dir."\\usuarios\\$usuario.rel";
                        open(ASOCIA,">$tRel");
                        for(my $i=0;$i<@relacion;++$i)
                        {
                              print ASOCIA "$relacion[$i]\n";
                        }
                        close ASOCIA;

                  # Se escriben los pesos
                        $tPes=$main::dir."\\usuarios\\$usuario.pes";
                        &art2::art2_guardar($tPes);
                  }
                  else
```

```
                  {
                  $main::txtSecundaria->insert('end',"¡Debe  ingresar  un
nombre!\n");
                        $main::txtSecundaria->see('end');
                        $main::ventanota->bell;
                        goto NOMBREMALO
                  }
            }

            print "presionado Aceptar $SegUsuario\n";
      }
      else
      {
      # Se ha cancelado la entrada del nombre de nuevo usuario.
            print "Cancelado\n";
      }
            } # Fin del else del botón siguiente, paso 5

            } # Fin de la rutina para el botón de comando $btnSiguiente
            ) # Fin de la config. de $btnSiguiente
                  ->grid(-column=>2,-row=>0);
      $vntDialogo->focus;
      $vntDialogo->protocol('WM_DELETE_WINDOW'=>sub{
                  $main::txtSecundaria->insert('end',"nusuario
destruida\n");
                  $main::btnNUsuario->configure(-state=>'normal');
                  $main::txtSecundaria->insert('end',"entreno
$main::entreno\n");
                  $main::txtSecundaria->see('end');
                  $vntDialogo->destroy;
                  $main::entreno='no';
                  $main::btnNUsuario->configure(-state=>'normal');} );
} #Fin de pasoA

#************************************

sub procesar
{
      $CumpleC="falso";
      $CumpleL="falso";
      $IenBlanco=&segmentar::CreaMatriz;
      &segmentar::Gemelo;          # Crea una matriz que es gemela a
GranMatriz

if ($paso==1)
{
# Esta rutina es llamada por el segmentador cuando se va a realizar una
segmentación en la
# etapa de entrenamiento, es muy parecida a &segmentar::procesar, con la
diferencia de que
# esta rutina sabe cuantos caracteres esperar y repite el proceso,
ajustando tx, ty y tol
# para obtener el resultado deseado

# Se establecen la cantidad de letras y líneas esperadas
      $CantLetras=17;
      $CantLineas=3;
```

```
# Ahora encuentro los parámetros a partir de los datos del tamaño de
imagen de segmentar
# dividiendolo entre el número de caracteres o líneas esperados
      $tx=($segmentar::Ultimox-$segmentar::Primerx)/9;
      $tx+=0.3*$tx;           # Aumento de un 30%
      $tx=int($tx);
      $ty=($segmentar::Ultimoy-$segmentar::Primery)/3;
      $ty+=0.05*$ty;          # Aumento de un 5%
      $ty=int($ty);
      $tolx=int($tx/10);            # El mínimo al inicio de tolx
      $tolx=$tolx<1?1:$tolx;
      $tolx=int($tolx);
      $toly=$ty*0.2;          # El minimo al inicio de toly
      $toly=$toly<1?1:$toly;
      $toly=int($toly);
      $main::txtSecundaria->insert('end',"tx $tx ty $ty tolx $tolx toly
$toly\n");
      $main::txtSecundaria->see('end');

} #fin del then del paso 1
elsif ($paso==2)
{
# Esta parte del código se activa la presionar procesar en el paso 2,
verifica que se encuentren
# las 27 letras mayúsculas
# Se establecen la cantidad de letras y líneas esperadas
      $CantLetras=27;
      $CantLineas=1;
      $main::txtSecundaria->insert('end',"Paso 2 tx $tx ty $ty tolx $tolx
toly $toly\n");
      $main::txtSecundaria->see('end');
      print "paso es $paso\n";

} #Fin del elsif del paso 2
elsif($paso==3)
{

# Se establecen la cantidad de letras y líneas esperadas
      $CantLetras=33;
      $CantLineas=1;
      $main::txtSecundaria->insert('end',"Paso 3 tx $tx ty $ty tolx $tolx
toly $toly\n");
      $main::txtSecundaria->see('end');
      print "paso es $paso\n";
} #Fin del elsif del paso 3
elsif($paso==4)
{
# Se establecen la cantidad de letras y líneas esperadas
      $CantLetras=31;
      $CantLineas=1;
      $main::txtSecundaria->insert('end',"Paso 4 tx $tx ty $ty tolx $tolx
toly $toly\n");
      $main::txtSecundaria->see('end');
      print "paso es $paso\n";

} # Fin del elsif del paso 4
```

```
else
{
# En caso de no ser uno de los números anteriores no hará nada

}

# Esta parte se corre para cualquiera de los pasos que llamen a la
función procesar

      if ($IenBlanco ne 'falso')
      {
      # Si la imagen esta en blanco detiene el proceso
            print "¡Imagen en blanco!\n";
            $main::txtSecundaria->insert('end',   "¡La   imagen   esta   en
blanco!\n");
            $main::txtSecundaria->see('end');
            $main::entreno='no';
            $vntDialogo->destroy;
      } # Fin de la verificación de la imagen en blanco

# Llamo a la función lazo para iniciar la segmentación, la cual llamará a
su vez a la función
# muestra para que el usuario visualice si le parece como ha quedado la
segmentación de la imagen
      &lazo;

} #Fin de procesar

#************************************

sub entrenar
{
$vntDialogo->focus;
my $paso=$_[0];

# Verifica que número de paso es, si es el paso 1 termina la función
if ($paso<=1)
{
      print "¡No pasa la tdc pues era el paso 1!\n";
      return;
}
else
{
# Como ya se realizó el conteo exitoso para el paso 2 se usa la TDC 2 y
aquí debería estar lo
# relacionado al entrenamiento de la red

# Llamo a la función para incrementar el tamaño de la letra proveniente
de la segmentación
# el resultado se almacenará en letras2.txt
      &zoom::zoom;
# Llamado a la función vecino8
# Nombre del archivo donde se encuentran los valores a transformar
      $archivo="letras2.txt";
# Cantidad de valores que deseo de regreso
      $n=$nEnt;
      &vecino8::tdcm($archivo,$n);
```

```
# Ahora el archivo destino.txt contiene los coeficientes a ser evaluados
por la art2
      open(COEF,"destino.txt");

while($linea=<COEF>)
{
# Quito el caracter de fin de línea
      chomp($linea);
# Leo el archivo DESTINO salto posibles caracteres de espacio y de fín de
línea, simplemente son
# ignorados
      if (($linea eq "Intro") or ($linea eq espacio) or ($linea eq ""))
      {
            next;
      } # Fin del if para descartar datos no válidos

# Si es una línea de datos válida procedo a dividirla, llamar la art y
asociar su salida con
# la letra
      @MatLinea=split(/ /,$linea);
      &art2::art2_1(@MatLinea);
      $resultado=&art2::art2_procesa;
# Con el número de la NG asigno el caracter a la matriz relación, con
ello en la pos. de
# @relacion se tiene la NG y en su celda la letra a la que esta asignada
      $relacion[$resultado]=$caracteres[$posCar];
# Aumento en 1 la cuenta de la pos en la matriz caracteres
      ++$posCar;

} # Fin del while de lectura de datos de destino.txt

# Cierro el archivo que contiene los coeficientes
      close COEF;
} # Fin del else para verificar que no sea el paso 1
# Le paso el enfoque a sig en la ventana de diálogo de la creación de
nuevos usuarios
      #$vntDialogo->focus;
      $vntDialogo->bell;
print "Termino entrenar\n";
} #Fin de entrenar

#************************************

sub muestra
{
# Rutina usada para mostrar el resultado de una segmentación
      $ventanota=$vntDialogo->Toplevel();
      $lblTexto=$ventanota->Label(-text=>"Se   han   segmentado   $letras
letras y $lineas líneas.\nRECOMENDACION: $nc y $nl. Cualquier consulta
acerca de los parámetros del segmentador puede hacerse en la ayuda del
programa.",-wraplength=>500)->pack(-anchor=>'n');
      $ventanota->configure(-title=>'Resultado de la segmentación');
# Aquí irían los cuadros de opción o entradas para variar los parámetros
de segmentación
      $frmParametros=$ventanota->Frame()->pack(-anchor=>'n');
      $lblTx=$frmParametros->Label(-text=>"Tx")->grid(-column=>0,-
row=>1);
```

```
      $entTx=$frmParametros->Entry()->grid(-column=>1,-row=>1);
      $btnTxd=$frmParametros->Button(-text=>'-',-command=>sub{&Txd})-
>grid(-column=>2,-row=>1);
      $btnTxa=$frmParametros->Button(-text=>'+',-command=>sub{&Txa})-
>grid(-column=>3,-row=>1);
      $lblTy=$frmParametros->Label(-text=>"Ty")->grid(-column=>0,-
row=>2);
      $entTy=$frmParametros->Entry()->grid(-column=>1,-row=>2);
      $btnTyd=$frmParametros->Button(-text=>'-',-command=>sub{&Tyd})-
>grid(-column=>2,-row=>2);
      $btnTya=$frmParametros->Button(-text=>'+',-command=>sub{&Tya})-
>grid(-column=>3,-row=>2);
      $lblTolx=$frmParametros->Label(-text=>"Tol    x")->grid(-column=>0,-
row=>3);
      $entTolx=$frmParametros->Entry()->grid(-column=>1,-row=>3);
      $btnTolxd=$frmParametros->Button(-text=>'-',-command=>sub{&Tolxd})
            ->grid(-column=>2,-row=>3);
      $btnTolxa=$frmParametros->Button(-text=>'+',-command=>sub{&Tolxa})
            ->grid(-column=>3,-row=>3);
      $lblToly=$frmParametros->Label(-text=>"Tol    y")->grid(-column=>0,-
row=>4);
      $entToly=$frmParametros->Entry()->grid(-column=>1,-row=>4);
      $btnTolyd=$frmParametros->Button(-text=>'-',-command=>sub{&Tolyd})
            ->grid(-column=>2,-row=>4);
      $btnTolya=$frmParametros->Button(-text=>'+',-command=>sub{&Tolya})
            ->grid(-column=>3,-row=>4);
      $entTx->insert("0.0","$tx");
      $entTy->insert("0.0","$ty");
      $entTolx->insert("0.0","$tolx");
      $entToly->insert("0.0","$toly");
# Creo frame porque posee scroll

$texto=$ventanota->Scrolled('Pane',-scrollbars=>'osoe')
                  ->pack(-expand=>'yes',-fill=>'both');
# Creo la etiqueta donde aparece la imagen de lo leído del archivo letras
$lblImagen=$texto->Label()->pack();

# Configuro un objeto Photo adentro de la etiqueta para poder dibujar en
ella
$imagen=$lblImagen->Photo();
$lblImagen->configure(-image=>$imagen);

# Lee el archivo letras.txt para mostrar su contenido en una imagen
open(FUENTE,"letras.txt") or die "No se pudo abrir letras.txt";
my $i=0;          # Desplazamiento en x
my $j=0;          # Desplazamiento en y
my @MatLetra=();  # Matriz donde se almacenan los datos de las letras
my @MatLinea=();  # Matriz en que se almacenan las líneas temporalmente
my $M=0;          # Número de lìneas
my $N=0;          # Número de columnas
my $CuentaPixeles=0;    # Cantidad de pixeles encontrados

# Leo línea por línea para obtener los caracteres que forman la imagen de
la letra
while ($hilera=<FUENTE>)
{
      chomp($hilera);
```

```
# Lee el dato para variar los parámetros de i y j si es espacio o intro
      if ($hilera eq 'Intro')
      {
      # Es un cambio de línea aumento $j en $ty
            $j+=$ty;
      # Se reinicia $i
            $i=0;
      }
      elsif ($hilera eq 'espacio')
      {
      # Es un espacio, aumento $i en $tx
            $i+=$tx;
      }
      elsif ($hilera eq 'letra')
      {
      # Se ha encontrado el final de una letra por lo que se imprime en
la etiqueta
            $N=$CuentaPixeles/$M;
            for (my $it=0;$it<$M;++$it)
            {
                  for (my $jt=0;$jt<$N;++$jt)
                  {
                        $imagen->put("$MatLetra[$it][$jt]",-
to=>($jt+$i,$it+$j));
                  }
            }
      # Asigno nuevos valores a las variables de control
            $CuentaPixeles=0;
            $M=0;
            $N=0;
            @MatLinea=();
            @MatLetra=();
            $i+=$tx;

      }
      else
      {
      # Lo que se tiene es una hilera de datos por lo que se agrega a la
matriz
            $hilera=~s/1/black /g;        # Cambio 1 por black
            $hilera=~s/0/white /g;        # Cambio 0 por white
            @MatLinea=split(/ /,$hilera);
            push @MatLetra,[@MatLinea];
            $CuentaPixeles+=@MatLinea;
            ++$M;
      }
}
      $main::txtSecundaria->insert('end',"Se   están   mostrando:   lineas
$lineas letras $letras tolx $tolx toly $toly\n");
            $main::txtSecundaria->see('end');
      print "Muestra dice: tx $tx ty $ty tolx $tolx toly $toly\n";
# Frame para los botones aceptar el resultado, ayuda etc.
$frmBot=$ventanota->Frame()->pack(-anchor=>'s');
$btnC=$frmBot->Button(-text=>'Cancelar el proceso',-command=>sub{
      $ventanota->destroy;
      $segmentar::ventanota->destroy();
      $vntDialogo->focus;
```

```
      })->grid(-column=>0,-row=>0);
# Este boton pide otra iteración de la imagen
$btnOtraIteracion=$frmBot->Button(-text=>'Realizar    otra    iteración',-
command=>sub{
      $tx=$entTx->get;  #Obtiene los datos de Tx
      $tx=($tx<1)?1:$tx;
      $tx=int($tx);
      $ty=$entTy->get;  # datos de ty
      $ty=($ty<1)?1:$ty;
      $ty=int($ty);
      $tolx=$entTolx->get;    # datos de tolx
      $tolx=($tolx<1)?1:$tolx;
      $tolx=int($tolx);
      $toly=$entToly->get;    # datos de toly
      $toly=($toly<1)?1:$toly;
      $toly=int($toly);
      $ventanota->destroy;
      &lazo;})->grid(-column=>1,-row=>0);
# Se acepta el resultado además si la segmentación fue correcta aparece
habilitado
      $btnS=$frmBot->Button(-text=>'Acepto        el        resultado',-
state=>'disabled',-command=>sub{
      $ventanota->destroy;
#Me aseguro que existe la venta de segmentación antes de destruirla
      #if($segmentar::ventanota ne "")
      #{$segmentar::ventanota->destroy;}
      $segmentar::ventanota->destroy;
      # Al cumplirse la condición llama a la función entrenar que realiza
la tdc y el entreno
      # de la red art con los valores de la transformada
      &entrenar($paso);
print "Ya se llamo a entrenar desde muestra\n";
      $vntDialogo->focus;
      $btnSiguiente->focus;
print "Se trato de pasar el enfoque\n";})->grid(-column=>2,-row=>0);
if(($CumpleC eq 'verdadero') and ($CumpleL eq 'verdadero'))
{
      $btnS->configure(-state=>'normal');
}
else
{
      $btnS->configure(-state=>'disabled');
}
# Le paso el enfoque al botón otra iteración
$btnOtraIteracion->focus;
#Llama a la ayuda en main
$btnAyuda=$frmBot->Button(-text=>'Ayuda',-command=>sub{&main::ayudaP;
&ayuda::Tema(3);})
      ->grid(-column=>3,-row=>0);
}
#***********************************
sub lazo
{
# Realiza la segmentación
# Este while se encarga de llamar a recortar las veces que sea necesaria,
limitado, y verificando
# si se ha encontrado el número de líneas y letras esperado
```

```
      #Llamado a la función de segmentación

      ($letras,$lineas)=&segmentar::recortar($segmentar::Primerx,$segment
ar::Primery,$segmentar::Ultimox,$segmentar::Ultimoy,$tx,$ty,$tolx,$toly);
            &segmentar::UsarGemelo;
            #Imprimo el número de letras y líneas que se ha encontrado
            $main::txtSecundaria->insert('end',"Se encontraron $letras
letras y $lineas líneas en la vuelta $Vuelta\n");
            $main::txtSecundaria->see('end');
      # Este If controla la cantidad de letras o caracteres encontrados
            if ($letras==$CantLetras)
            { # Si tiene éxito informa que la cantidad de letras esta
bien
                  $main::txtSecundaria->insert('end',"¡Cant de Let
bien!\n");
                  $main::txtSecundaria->see('end');
                  $CumpleC='verdadero';
            # nc es la recomendación en lo que respecta a las letras
                  $nc="El número de LETRAS esta bien por tanto solo en
caso de ser muy necesario se recomienda variar el Tamaño de la Ventana en
x y la Tolx, por ejemplo si el dibujo de las letras segmentadas muestra
que ha segmentado mal algún caracter";
                  print "Cant de let bien\n";
            }
            else
            {
                  $CumpleC='falso';
                  if(($letras-$CantLetras)<0)
                  {
                        $nc='La cantidad de LETRAS encontradas es MENOR
que lo esperado se recomienda revisar el dibujo que muestra el resultado
de lo segmentado para saber si es el tamaño y/o tolerancia en X y/o el
tamaño y/o tolerancia en Y el causante de que no se encuentre el número
de letras correcto. Usualmente es necesario disminuir TX.';
                  }
                  else
                  {
                        $nc='La cantidad de LETRAS encontradas es MAYOR
que lo esperado se recomienda revisar el dibujo que muestra el resultado
de lo segmentado para saber si es el tamaño y/o tolerancia en X y/o el
tamaño y/o tolerancia en Y es el causante de que no se encuentre el
número de letras correcto. Usualmente es necesario aumentar TX.';
                  }
            }
      # Este If controla la cantidad de líneas encontradas
            if ($lineas==$CantLineas)
            { # Si tiene éxito ya no varía ty
                  $main::txtSecundaria->insert('end',"¡Cant de Lin
bien!\n");
                  $main::txtSecundaria->see('end');
                  $CumpleL='verdadero';
                  $nl="el número de LINEAS esta bien por tanto solo en
caso de ser muy necesario se recomienda variar el Tamaño de la Ventana en
Y y la Toly, por ejemplo si el dibujo de las letras segmentadas muestra
que ha segmentado mal algún caracter";
                  print "Cant de Lin bien\n";
```

```
            }
            else
            {
                  $CumpleL='falso';
                  if(($lineas-$CantLineas)<0)
                  {
                        $nl='la cantidad de LINEAS encontradas es MENOR
que lo esperado se recomienda revisar el dibujo que muestra el resultado
de lo segmentado para saber si es el tamaño y/o tolerancia en Y y/o el
tamaño y/o tolerancia en X el causante de que no se encuentre el número
de LINEAS correcto. Usualmente es necesario disminuir TY.';
                  }
                  else
                  {
                        $nl='la cantidad de LINEAS encontradas es MAYOR
que lo esperado se recomienda revisar el dibujo que muestra el resultado
de lo segmentado para saber si es el tamaño y/o tolerancia en Y y/o el
tamaño y/o tolerancia en X el causante de que no se encuentre el número
de letras correcto. Usualmente es necesario aumentar TY.';
                  }
            }


# Este if se encarga de comprobar que se han alcanzado los parámetros
buscados, si es cierto
# entonces activa el botón sig del paso1

      if (($CumpleC eq 'verdadero') and ($CumpleL eq 'verdadero'))
      {

      # Se habilita el botón sig.
            $btnSiguiente->configure(-state=>'normal');
      # Llamo a la función muestra para que presente el cuadro de diálogo
de si el usuario
      # aceptará esa segmentación
            &muestra;
      }
      else
      {
      # Se deshabilita el botón sig.
            $btnSiguiente->configure(-state=>'disabled');
      # Llamo a la función muestra
            &muestra;
      }
      $vntDialogo->bell;

}
#************************************
sub Txd
{
      $tx=$entTx->get;
      --$tx;
      $tx=int($tx);
      $tx=($tx<1)?1:$tx;
      $entTx->delete('0.0','end');
      $entTx->insert('0.0',"$tx");
}
```

```
sub Txa
{
      $tx=$entTx->get;
      ++$tx;
      $tx=int($tx);
      $entTx->delete('0.0','end');
      $entTx->insert('0.0',"$tx");
}

sub Tyd
{
      $ty=$entTy->get;
      --$ty;
      $ty=int($ty);
      $ty=($ty<1)?1:$ty;
      $entTy->delete('0.0','end');
      $entTy->insert('0.0',"$ty");
}

sub Tya
{
      $ty=$entTy->get;
      ++$ty;
      $ty=int($ty);
      $entTy->delete('0.0','end');
      $entTy->insert('0.0',"$ty");
}

sub Tolxa
{
      $tolx=$entTolx->get;
      ++$tolx;
      $tolx=int($tolx);
      $entTolx->delete('0.0','end');
      $entTolx->insert('0.0',"$tolx");
}

sub Tolxd
{
      $tolx=$entTolx->get;
      --$tolx;
      $tolx=int($tolx);
      $tolx=($tolx<1)?1:$tolx;
      $entTolx->delete('0.0','end');
      $entTolx->insert('0.0',"$tolx");
}

sub Tolya
{
      $toly=$entToly->get;
      ++$toly;
      $toly=int($toly);
      $entToly->delete('0.0','end');
      $entToly->insert('0.0',"$toly");
}
```

```
sub Tolyd
{
      $toly=$entToly->get;
      --$toly;
      $toly=int($toly);
      $toly=($toly<1)?1:$toly;
      $entToly->delete('0.0','end');
      $entToly->insert('0.0',"$toly");
}
1; # Valor de retorno
```

A.4.4. *reglas*

```
# Rutina para encontrar errores en el reconocimiento
# La primera parte de la función corrige
# Revisado: 27/02/04
package reglas;
sub corrige
{
# USANDO EXPRESIONES REGULARES PARA LOS CAMBIOS FACILES
      $hilera=$main::txtPrincipal->get('0.0','end');
# Para cuando aparece un uno justo antes de una letra, debe ser 1
      $hilera=~s/1[a-zA-Z]/l$&/g;
      $hilera=~s/l1/l/g;
# Si aparece: punto espacio letra minúscula la cambia a mayúscula
      $hilera=~s/\. [a-z]/\U$&/g;
# Cuando aparece minúscula mayúscula unidas cambia todo a minúsculas
      $hilera=~s/[a-z][A-Z]/\L$&/g;
      $hilera=~s/[á-ú][A-Z]/\L$&/g;
      $hilera=~s/[a-z][A-Z]/\L$&/g;
      $hilera=~s/[á-ú][A-Z]/\L$&/g;
      $hilera=~s/[a-z][A-Z] /\L$&/g;
# Una ñ al final de una palabra
      $hilera=~s/ñ\n/n\n/g;
      $hilera=~s/Ñ\n/N\n/g;
# Una c al principio de una palabra y no es r la consonante sig. es e
      $hilera=~s/(\W)(c)([^ a,e,i,o,u,r,l])/$1e$3/g;
# Un Gos debe ser Cos
      $hilera=~s/(\W)Gos/$1Cos/g;
      $hilera=~s/(\W)gos/$1cos/g;
# Para cuando aparece un uno justo después de una letra, debe ser l
      $hilera=~s/([a-zA-Z])1/$1l/g;
# Una q antes de cualquier letra que no sea u debe ser g
      $hilera=~s/(q)([^ u,U])/g$2/g;
      $hilera=~s/(Q)([^ u,U])/O$2/g;
# Una s entre dos consonantes debe ser a
      $hilera=~s/([^ a,e,i,o,u])(s)([^ a,e,i,o,u,\W])/$1a$3/g;
# #.o,O,c,C por #.0, la o por cero
      $hilera=~s/(\d)(\.)([o,O,c,C])/$1.0/g;
# Una s después de una consonante al final de una palabra debe ser a
      $hilera=~s/([^ a,e,i,o,u])(s)\b/$1a/g;
# Una g al final de una palabra debe ser a
      $hilera=~s/(\w)(g)\b/$1a/g;
# Una p entre dos consonantes debe ser o
      $hilera=~s/([^ a,e,i,o,u,\W])(p)([^ a,e,i,o,u])/$1o$3/g;
```

```
# un ¡ entre dos letras debe ser i
      $hilera=~s/([a-zA-Z])(¡)([a-zA-Z])/$1i$3/g;
# Una v entre dos consonantes debe ser u
      $hilera=~s/([^ a,e,i,o,u])(v)([^ a,e,i,o,u])/$1u$3/g;
      $hilera=~s/([^ A,E,I,O,U])(V)([^ A,E,I,O,U])/$1U$3/g;
      $hilera=~s/([^ a,e,i,o,u])([^ a,e,i,o,u])(v)/$1$2u/g;
      $hilera=~s/([^ A,E,I,O,U])([^ A,E,I,O,U])(V)/$1$2U/g;
# Un 9 antes de una u debe q, antes de otra letra debe ser g
      $hilera=~s/(9)([^ \d,\.,\W,a,e,i,o])/q$2/g;
      $hilera=~s/(9)([^ \d,\.,\W])/g$2/g;
# Un $ con un caracter de letra alrededor debe ser s
      $hilera=~s/(\$)([^ \d,\.,\W])/s$2/g;
      $hilera=~s/(\w)(\$)/$1s/g;
# Esta parte es para ingresar el resultado de los cambios fáciles en la
ventana de texto
      &main::BorrarTodo;
      $main::txtPrincipal->insert('0.0',$hilera);
}
1;
```

A.4.5 *segmentar*

```
#Ver 4.0
#Segmenta varios caracteres y los imprimi en un archivo de texto,
#Revisado 26/02/04
#Autor: Yo - M

#Módulo para segmentar imágenes
#Deberá recibir la dir del archivo de imagen a mostrar y sacará
divisiones de la imagen donde
#supone debera existir una letra a identificar
#Pasos:
# 1- Toma la imagen y la muestra en una etiqueta, función &MostrarI()
# 2- Pasa la inf. de la imagen a una matriz de ancho x alto, con los
datos 1 (negro) ó 0,
# 3- Encuentra los primeros y últimos pixeles con inf. (color), para
formar un marco que será el
# punto de partida para el muestreo
# 4- Comienza a muestrear el marco para dividir la imagen:
# 4.1- Muestrea de arriba hacia abajo y de izq. a der. para encontrar los
pixeles con color
# 4.2- Cuando encuentra los pixeles los pasa a otra matriz más pequeña
verificando siempre su
# conexión con los otros pixeles a través de la "cruz de conexión"
(vecino 4 extendido) con un
# valor de tolerancia (separación) dado.
# 4.3- Realiza la búsqueda adentro de los tamaños de ventana máximos
dados
# 4.4- Una vez muestreada toda la ventana, se borran de la imagen
original todos los datos que
# han sido pasados a la nueva mini matriz
# 4.5- Repite el proceso hasta llegar a los límites de GranMatriz


#Funciones
```

```
# &MostrarI(archivo de imagen)      Muestra la imagen en una etiqueta
# &CreaMatriz                 Crea la matriz con los datos de la imagen.
Llama a las sig 4 f:
# &PrimerP_y                  Encuentra 1° pixel en y
# &PrimerP_x                  Encuentra 1° pixel en x
# &UltimoP_y                  Encuentra último pixel en y
# &UltimoP_x                  Encuentra último pixel en x
# &recortar(x1,y1,x2,y2,tx,ty,Tolerancia)      Divide   la   imagen   en
pequeños cuadros, donde se
#                       supone hay una letra,
# &procesar             Se encarga de llamar a recortar si la imagen no
esta en blanco
#                       y muestra el archivo de letras obtenido en la
ventana ppal.


package segmentar;
#_____________________________________________________________________
_____________________________
#                       SUBRUTINAS
#_____________________________________________________________________
_____________________________

sub MostrarI()
{ #Recibe un argumento, el nombre del archivo a mostrar, el cual debe ser
bmp o gif
      $ventanota =$main::ventanota->Toplevel();
      $marco  =  $ventanota->Scrolled('Pane',-scrollbars=>'osoe')->pack(-
expand=>'yes',-fill=>'both');
      $etiqueta=$marco->Label()->pack;   #  es  la  etiqueta  es  donde
aparece la imagen
      $cuadroB=$ventanota->Frame()->pack; # es el frame para los botones
      $ImagenAMostrar=$_[0];
      $ventanota->configure(-title=>"Imagen: $ImagenAMostrar");
#     print "imagen a mostrar $ImagenAMostrar $tx $ty $tol\n";
      $imagen=$etiqueta->Photo(-file=>$ImagenAMostrar); #aquí aparece el
nombre del archivo
      $etiqueta->configure(-image=>$imagen);   # pongo la imagen en la
etiqueta
      #esto es temporal para poner un botón en la imagen
      if ($main::entreno eq 'no')
      { # Si no hay entrenamiento ejecuto de esta manera
            $main::txtSecundaria->insert('end',"valor      de      entreno
$main::entreno\n");
            $botonB=$cuadroB->Button(-text=>'Procesar Imagen',
                  -command=>\&procesar)->grid(-row=>0,-column=>0);

      }
      else
      {
            $main::txtSecundaria->insert('end',"valor      de      entreno
$main::entreno\n");
            $botonB=$cuadroB->Button(-text=>'Procesar Imagen',
                  -command=>\&nusuario::procesar)->grid(-row=>0,-
column=>0);
      }
```

```
      $botonC=$cuadroB->Button(-text=>"Cancelar",-command=>sub{
$ventanota->destroy;})
      ->grid(-row=>0,-column=>1);
      $ventanota->focus;      #le paso el enfoque para que pueda activar
el botón de procesar
      $botonB->focus;         # Enfoque al botón procesar


} #Fin de MostrarI

#-------------------------------------------------------------------------
----------------------

sub CreaMatriz
{
      print "trabajando creamatriz\n";
      #Se inicializa la matriz a trabajar
      @GranMatriz=();
      $alto=$imagen->height;
      $ancho=$imagen->width;
#     print "ancho $ancho alto $alto\n";
      for ($y=0;$y<=($alto-1);++$y)
      {
            for ($x=0;$x<=($ancho-1);++$x)
            { #No pongo la asignación directamente debido a que get
regresa los 3 componentes
            #de color del pixel
                  @colordepixel=$imagen->get($x,$y);
                  $GranMatriz[$x][$y]=$colordepixel[0]>0?0:1;
#                 print "$GranMatriz[$x][$y] ";
            }
#           print "\n";
      }
      print "Se terminó de crear GranMatriz\n";
      $Primery=&PrimerP_y;
      if ($Primery eq "imagen en blanco")
      {
            return("verdadero");
      }
      $Primerx=&PrimerP_x;
      $Ultimoy=&UltimoP_y;
      $Ultimox=&UltimoP_x;
      #imprime los valores obtenidos
#     print "Primer y $Primery ";
#     print "Primer x $Primerx ";
#     print "Ultimo y $Ultimoy ";
#     print "Ultimo x $Ultimox\n";
      #Regresa que imagen en blanco es falso, por tanto si hay imagen a
examinar
      return ("falso");
} #Fin de CreaMatriz

#-------------------------------------------------------------------------
----------------------

sub PrimerP_y()
```

```
{ #Encuentra la posición del primer pixel en negro en un escaneo de
arriba hacia abajo de izq a
# derecha, regresa 1 escalar
      my $y=0;
      my $encontrado="falso";
      my $x=0;
      while ($encontrado eq "falso" and $y<=($alto-1))
      {
            $x=0;
            while ($encontrado eq "falso" and $x<=($ancho-1))
            {
                  $encontrado=$GranMatriz[$x][$y]==0?"falso":"verdadero";
                  ++$x;
            }
            ++$y;
      }
      if ($encontrado eq "falso")
      {
            return ("imagen en blanco");
      }
      else
      {
            my $Py=$y-1;
            return($Py);
      }

} #Fin de PrimerP_y

#-------------------------------------------------------------------------
----------------------

sub PrimerP_x()
{ #Se encuentra la pos de la primera columna con letras, regresa 1
escalar
      my $x=0;
      my $encontrado="falso";
      my $y=0;
      while ($encontrado eq "falso" and $x<=($ancho-1))
      {
            $y=0;
            while ($encontrado eq "falso" and $y<=($alto-1))
            {
                  $encontrado=$GranMatriz[$x][$y]==0?"falso":"verdadero";
                  ++$y;
            }
            ++$x;
      }

      return($x-1);

}
#-------------------------------------------------------------------------
----------------------

sub UltimoP_y()
{
      my $x=($ancho-1);
```

```
	my $y=($alto-1);
	my $encontrado="falso";
	while ($encontrado eq "falso" and $y>=0)
	{
		$x=($ancho-1);
		while ($encontrado eq "falso" and $x>=0)
		{
			$encontrado=$GranMatriz[$x][$y]==0?"falso":"verdadero";
			--$x;
		}
		--$y;
	}

	return ($y+1);
} #Fin UltimoP_y()
#-------------------------------------------------------------------------
----------------------

sub UltimoP_x()
{
	my $x=($ancho-1);
	my $y=($alto-1);
	my $encontrado="falso";
	while ($encontrado eq "falso" and $x>=0)
	{
		$y=($alto-1);
		while ($encontrado eq "falso" and $y>=0)
		{
			$encontrado=$GranMatriz[$x][$y]==0?"falso":"verdadero";
			--$y;
		}
		--$x;

	}
	return($x+1);

} #Fin UltimoP_x()
#-------------------------------------------------------------------------
----------------------

sub recortar()
{ #Esta es la parte ppal. se reciben 7 escalares como argumento
	my $x1=$_[0];		# 1° punto en x
	my $y1=$_[1];		# 1° punto en y
	my $x2=$_[2]+1;		# Último punto en x
	my $y2=$_[3]+1;		# Último punto en y
	my $tx=$_[4];		# Ancho de la ventana para segmentar
	my $ty=$_[5];		# Altura de la ventana para segmentar
	my $Tolx=$_[6];		# Anchura de la "cruz de conexión"
	my $Toly=$_[7];		# Altura de la "cruz de conexión"
	my $fin = "falso";	# Dice cuando se ha terminado de muestrear
el "marco"
	my $i;
	my $j;
	my $menorx=$x1;		# Pos. de la ventana en x inicial en
GranMatriz
```

```
      my $menory=$y1;         # Pos. de la ventana en y inicial en
GranMatriz
      my $minx;          # Pos. EN la ventana donde se encontró el primer
pixel con color
      my $miny;          # Pos. EN la ventana donde se encontró el primer
pixel con color
      my $maxx;          # Pos. EN la ventana donde se encontró el último
pixel con color
      my $maxy;          # Pos. EN la ventana donde se encontró el último
pixel con color
      my $Mminx=$x1;          # Pos. en GranMatriz donde se encontró el
primer pixel de ese
                         # muestreo.
      my $Mminy;         # Pos. en GranMatriz donde se encontró el primer
pixel de ese
                         # muestreo.
      my $dato;          # Se almacen el valor de un pixel de la imagen
original que ahora
                         # se encuentra en GranMatriz
      my $Pdato="verdadero";  # Indica si el dato que se esta tomando es
el primero de su
                         # ventana para saltarse la cruz de muestreo
      my @ventana=();         # Se limpia la ventana donde se almacenará
el recorte
      my $conectado;          #  Indica si el pixel encontrado esta
conectado a los demás
      my $vx;                 # Pos. de un pixel en la ventana ($i-
$menorx)
      my $vy;                 # Pos. de un pixel en la ventana ($j-
$menory)
      my $maxi;          #  Pos. max que recorre en x el muestreo en
GranMatriz
      my $maxj;          #  Pos. max que recorre en y el muestreo en
GranMatriz
      my $LineaB;        # Línea en blanco, para pasar a la sig. línea de
GranMatriz
      my $ColumnaB;           # Columna en blanco, para pasar a la sig.
col. de GranMatriz.
      my $CuentaPix;          # Cuenta la cantidad de pixeles que se han
encontrado, si son
                         # muy pocos rechaza el caracter por tratarse solo
de basura
      my $letras;        # Es el número de caracteres que se han encontrado
      my $lineas;        # Número de lìneas que se ha encontrado
      my $posibleL;           # Indica cuando hay una posible linea pero
es necesario verificar
                         # la cuenta de pixeles
      my $OrigMenory=$y1;     # Guardo el valor original de menory antes
de que sea afectado
                         # el lazo PREVIOQC
      my $nada="verdadero";   # Indica si se encontró un pixel con color
en el lazo PREVIOQC

#LINEAS TEMPORALES PARA IMPRIMIR LAS LETRAS EN UN ARCHIVO DE TEXTO
open(LETRAS,">letras.txt");
#LINEAS TEMPORALES PARA IMPRIMIR LAS LETRAS EN UN ARCHIVO DE TEXTO
print "Toleranciax $Tolx\n";
```

```
      $letras=0;
      $lineas=1;

LAZOW:      while ($fin eq "falso")
      { #Limpio ventana
            @ventana=();            # Limpio la ventana para borrar el
dato anterior
            $posibleL="falso";
            $nada="verdadero";
      #Limpio algunas variables de control
            $CuentaPix=0;           # Se reinicia la cuenta de pixeles en
cada caracter
            $maxx=0;
            $maxy=0;
      # Este lazo es para hacer un muestreo de izq a der de arriba hacia
abajo para encontrar un
      # pixel con color entre Mminx y Mminx+tx, al encontrarlo varia
menory a la pos en y donde
      # se encuentra el pixel encontrado
            $menory=$OrigMenory;
PREVIOQC:   for (my $j=$OrigMenory;$j<($OrigMenory+0.8*$ty);++$j)
            {
                  if ($j>$y2)
                  {
                  # Si j es mayor que $y2 entonces ya se pasó del tamaño
de la matriz que
                  # contiene la imagen por lo que termina este lazo
                  #print "Llegó al final de la matriz en PREVIOQC\n";
                        last PREVIOQC;

                  }
                  # Realiza una lazo que busca de izq a der por un pixel
con color, si lo
                  # encuentra termina PREVIOQC y asigna a $menory el
valor que se encuentra
                  # en j
PREVIOQCX:        for (my $i=$Mminx;$i<=($Mminx+1.7*$tx);++$i)
                  {
                  # Busca un pixel con color
                        if($i>$x2)
                        {
                        # Superó el ancho de la matriz, se hace la sig
iteración
                              last PREVIOQCX;
                        }
                        if ($GranMatriz[$i][$j]!=0)
                        {
                        # Encontro un pixel con color por lo tanto termina
PREVIOQC y
                        # cambia el valor de menory
                              #print "Encontró un pixel con color en
$j\n";

                              $menory=$j;
                              $nada="falso";
                              last PREVIOQC;
                        }
                  }
```

```
                # No encontró un pixel con color en esa línea aumenta
la cuenta en j
            }
      # Revisa si se ha encontrado un pixel con color en PREVIOQC
            # print "nada es $nada .";
            if ($nada eq "verdadero")
            {
            #Cambio el valor de $Mminx a $Mminx+tx, pues el próximo lazo
perdería tiempo
            # buscando en donde ya se buscó y se realiza otra iteración
de PREVIOQC
                  $menorx=$menorx+$tx;
                  if ($menorx<$x2)
                  {
                  # Si no se pudo encontrar un caracter con color y no se
ha llegado al
                  # final de la lína entonces se realiza otra iteración
en PREVIOQC pero
                  # con Mminx variado
                        goto PREVIOQC;
                  # De lo contrario ya s superó el ancho y se deja que
pase al sig lazo
                  # que haría que se pasase la búsqueda de nueva línea
pues se ha
                  # superado el ancho de la matriz
                  }
                  # Imprimo un espacio en el archivo letras ya que se
pasó la ventana sin
                  # encontrar un pixel con color

                  print LETRAS "espacio\n";
            }
            if ($nada eq "falso")
            {
            # Encontró un pixel con color, por lo que disminuye el valor
de
            # menory
                  $menory=$menory-0.1*$ty;

            }

            #Desplazo el muestreo en x a la primera pos., a partir de la
primera pos. donde
            #se encontró un pixel con color, en donde aparezca un pixel
con color
            $ColumnaB="verdadero";
QUITAC:           for (my $Descartax=$Mminx;$Descartax<=$x2;++$Descartax)
            { #Busco un pixel con color para ubicar esa pos en x como
$menorx
QUITACL:          for   (my   $Descartay=$menory;$Descartay<=($menory+$ty-
0.2*$ty);++$Descartay)
                  {
                        if ($Descartay>$y2)
                        {  #Si  ya  se  llegó  a  la  parte  inferior  de
GranMatriz
                        #se evalúa la sig pos. de x
                              last QUITACL;
```

```
                        }

      $ColumnaB=$GranMatriz[$Descartax][$Descartay]==0?"verdadero":"falso
";
                        if ($ColumnaB eq "falso")
                        { #Se encotró un pixel con color
                              $menorx=$Descartax;
                              $Pdato="verdadero";
#                             print "Pos x $Descartax ";
                              last QUITAC;
                        }

                  } #Fin de QUITACL
                  if ($Descartax>=($menorx+$tx))
                  { #Espacio en blanco
                        #print "se encontró un espacio\n";
                        $menorx+=$tx;
                        print LETRAS "espacio\n";
                  }
            } #Fin de QUITAC
            if ($ColumnaB eq "verdadero")
            { #Se terminó de examinar toda la línea y es necesario pasar
a otra línea, además
            # se aumenta el conteo de $lineas en uno
                  $menory=$ty+$menory;
                  #print "se pasó a otra línea\n";
            # Se cambia el valor de Mminx ya que no hay más datos en esa
línea y se comienza
            # a rastrear la próxima VERIFICAR EL PROGRAMA
                  $Mminx=$x1;
                   #Se pasa a la pos inicial en x de la sig. línea del
marco
                  $menorx=$x1;
                  #Examina toda la línea para descartarla si toda la
línea
                  #esta vacía
                  $LineaB="verdadero";
                  my $Descartary=$menory;
QUITAL:                 while     ($LineaB     eq     "verdadero"     and
$Descartary<=$y2)
                  {
                        for                                          (my
$Descartarx=$x1;$Descartarx<=$x2;++$Descartarx)
                        { #Si la pos en GranMatriz es cero entonces la
                        #linea en blanco es verdadera

      $LineaB=$GranMatriz[$Descartarx][$Descartary]==0?"verdadero":"falso
";
                              if ($LineaB eq "falso")
                              { #encontró un pixel con color por lo tanto
el
                              #nuevo valor de menory será:
                                    $Pdato="verdadero";
                                    $menory=$Descartary;
                                    $OrigMenory=$menory;
                                    $posibleL="verdadero";
```

```
                print "posible línea encontrada falta verificar si
tiene suficientes pixeles\n";
                # PRUEBA para que busque de nuevo en la linea la pos de
menorx
                                $ColumnaB="verdadero";
                                goto QUITAC;
                                last QUITAL;
                           }
                     }
#                    print "línea en blanco $Descartary descartada\n";
                     ++$Descartary;

                } #Fin de QUITAL
                if ($LineaB eq "verdadero")
                { #Si termino LineaB con verdadero es que ya rastreo
todas las
                #líneas y no hay datos por lo que termina superwhile
           print "Todas las demás líneas están en blanco letras $letras
lineas $lineas\n";
                     last LAZOW;
                }

           } #Fin del if ($ColumnaB eq "verdadero")
#************************************************************
#*PRIMERA PASADA DE ARRIBA HACIA ABAJO DE IZQUIERDA A DERECHA*
#************************************************************
LAZOX1:         for($i=$menorx;$i<=($menorx+$tx);++$i)
           {
                $maxi=$i;
                $vx=$i-$menorx;
                if ($i>$x2)
                { #Indica que $i ya se pasó del valor máximo del marco
de la
                #imagen, por lo tanto termina este for
                last LAZOX1;
                }


LAZOY1:                 for ($j=$menory;$j<=($menory+$ty);++$j )
                { #$menory + $ty es el desfase más el tamaño máximo que
puede tener la
                #ventana
                     $maxj=$j;
#                    print "j $j ";
                     if ($j>$y2)
                     { #Indica que $j ya se pasó del valor máximo del
marco de la
                     #imagen, por lo tanto termina este for
                     last LAZOY1;
                     }
                     #Se obtiene el valor del dato de GranMatriz
                     $dato=$GranMatriz[$i][$j];
                     $vy=$j-$menory;
#                    print "vy $vy ";
                     if ($dato==0)
                     {
                          $ventana[$vx][$vy]=$dato;
```

```
                              next LAZOY1;      #Otra iteración del for
más interno
                        }
                        elsif ($Pdato eq "verdadero")
                        { #Si el dato no es 0 y es el primero entonces
solo lo guarda
                        # en la matriz ventana y borra el original, no
verifica
                        # conexión con otros datos, además aumenta el
conteo de letras
                              $ventana[$vx][$vy]=$dato;
                              $GranMatriz[$i][$j]=0;
                              $Pdato="falso";
                              #Establecimiento 1 de mínimos y máximos
                              $maxx=$vx;
                              $maxy=$vy;
                              $minx=$vx;
                              $miny=$vy;
                              $Mminx=$i;
                              $Mminy=$j;

#                             print "Mminx  $Mminx Mminy $Mminy\n";
#                             print "vx $vx vy $vy\n";
                        }
                        else
                        { #El dato no es 0 y no es el primero por tanto se
pasa la
                        # "cruz de conexión"
                              $conectado="falso";

CRUZV1:                             for          (my          $cy=($vy-
$Toly);$cy<=($vy+$Toly);++$cy)
                                    {
                                    if ($cy<0)
                                    {
                                          next CRUZV1;
                                    }

CRUZH1:                                   for        (my        $cx=($vx-
$Tolx);$cx<=($vx+$Tolx);++$cx)
                                    {
                                          if ($cx<0)
                                          {
                                                next CRUZH1;
                                          }
                                          if($cx==$vx and $cy ==$vy)
                                          { #Esta es la pos. del pixel que
tiene
                                          #la pos. original
                                          #por lo que no se toma en cuenta
                                                next CRUZH1;
                                          }

      $conectado=$ventana[$cx][$cy]==1?"verdadero":"falso";
                                          if($conectado eq "verdadero")
                                          { #Hay un vecino por tanto ya no
                                          #hay necesidad
```

```
                                        #de seguir buscando
                                              last CRUZV1;
                                        }

                                  }
                            }

                            if ($conectado eq "falso")
                            { # Como el pixel no se encuentra conectado
con otro
                            # es ignorado y se pasa a la sig. iteración
de LAZOY
#                                 print "no conectado ";
                                  $ventana[$vx][$vy]=0;
                                  next LAZOY1;
                                        }
                            #si esta conectado!!!!!!!!!!!
                            $ventana[$vx][$vy]=$dato;
                            $GranMatriz[$i][$j]=0;
                            #Establecimiento 2 de mínimos y máximos
                            $maxx=$vx>$maxx?$vx:$maxx;
                            $maxy=$vy>$maxy?$vy:$maxy;
                            $minx=$vx<$minx?$vx:$minx;
                            $miny=$vy<$miny?$vy:$miny;
                            $Mminx=$i<$Mminx?$i:$Mminx;
                            $Mminy=$j<$Mminy?$j:$Mminy;
      # Aumento la cuenta de pixeles, de esta manera controlo si es
basura o inf.
                            ++$CuentaPix;
                        } #Fin del else grande 1
                  } #Fin del For 1 en y
            } #Fin del For 1 en x

#*************************************************************
#*DE ARRIBA HACIA ABAJO DE DERECHA A IZQUIERDA***************
#*************************************************************
LAZOX2:           for($i=($menorx+$tx);$i>=$menorx;--$i)
            {
                  $vx=$i-$menorx;
                  if ($i>$x2)
                  { #Indica que $i ya se pasó del valor máximo del marco
de la
                  #imagen, por lo tanto sigue en otra iteración ya que $i
desciende
                  next LAZOX2;
                  }

LAZOY2:                 for ($j=$menory;$j<=($menory+$ty);++$j )
                  { #$menory + $ty es el desfase más el tamaño máximo que
puede tener la
                  #ventana
#                       print "j $j ";
                        if ($j>$y2)
                        { #Indica que $j ya se pasó del valor máximo del
marco de la
                        #imagen, por lo tanto termina este for
                        last LAZOY2;
```

```
                        }
                        #Se obtiene el valor del dato de GranMatriz
                        $dato=$GranMatriz[$i][$j];
                        $vy=$j-$menory;
#                       print "vy $vy ";
                        if ($dato==0)
                        {
                              #$ventana[$vx][$vy]=$ventana[$vx][$vy];
                              next LAZOY2;      #Otra  iteración  del  for
más interno
                        }
                        else
                        { #El dato no es 0 por tanto se pasa la
                        # "cruz de conexión"
                              $conectado="falso";

CRUZV2:                             for           (my           $cy=($vy-
$Toly);$cy<=($vy+$Toly);++$cy)
                                    {
                                    if ($cy<0)
                                    {
                                          next CRUZV2;
                                    }

CRUZH2:                                   for         (my         $cx=($vx-
$Tolx);$cx<=($vx+$Tolx);++$cx)
                                    {
                                          if ($cx<0)
                                          {
                                                next CRUZH2;
                                          }
                                          if($cx==$vx and $cy ==$vy)
                                          { #Esta es la pos. del pixel que
tiene
                                          #la pos. original
                                          #por lo que no se toma en cuenta
                                                next CRUZH2;
                                          }

      $conectado=$ventana[$cx][$cy]==1?"verdadero":"falso";
                                          if($conectado eq "verdadero")
                                          { #Hay un vecino por tanto ya no
                                          #hay necesidad
                                          #de seguir buscando
                                                last CRUZV2;
                                          }

                                    }
                              }

                              if ($conectado eq "falso")
                              { # Como el pixel no se encuentra conectado
con otro
                              # es ignorado y se pasa a la sig. iteración
de LAZOY
#                             print "no conectado ";
                                    $ventana[$vx][$vy]=0;
```

```
                                    next LAZOY2;
                                        }
                                #si esta conectado!!!!!!!!!!
                                $ventana[$vx][$vy]=$dato;
                                $GranMatriz[$i][$j]=0;
                                #Establecimiento 3 de mínimos y máximos
                                $maxx=$vx>$maxx?$vx:$maxx;
                                $maxy=$vy>$maxy?$vy:$maxy;
                                $minx=$vx<$minx?$vx:$minx;
                                $miny=$vy<$miny?$vy:$miny;
                                $Mminx=$i<$Mminx?$i:$Mminx;
                                $Mminy=$j<$Mminy?$j:$Mminy;
            # Aumento la cuenta de pixeles, de esta manera controlo si es
basura o inf.
                                ++$CuentaPix;
                            } #Fin del else grande 2


                    }#Fin del for 2 de der a izq

            } #Fin del for 2 de arriba hacia abajo
#MITAD:
#Escape de cuadros que son demasiado pequeños, restos de imágenes
#son ignorados
      if ($CuentaPix<(0.8*($Toly+$Tolx)))
      {#Se salta la parte de imprimir en el archivo, o cualquier otra
cosa en el futuro,
      # para no tomar esos datos que solo serían residuos
            #print "escape de inf muy pequeña tiene $CuentaPix
pixeles\n";
            goto ESCAPE;
      }
      # Aumenta $letras en 1
      ++$letras;
      if ($posibleL eq "verdadero")
      {
            # Si se ha señalado que hay una posible linea se inserta el
caracter intro antes
            # de la letra sig.
            print "Era una línea relevante se inserta Intro y se aumenta
conteo de líneas\n";
            print LETRAS "Intro\n";
            ++$lineas;
      }

#     print "aumento de letras $letras\n";
#LINEAS DE IMPRESION DE LA VENTANA EN EL ARCHIVO LETRAS
      #print "pos de la ventana en la matriz $menorx y $menory Mmin
$Mminx y $Mminy\n";
            for (my $Pj=$miny;$Pj<=$maxy;++$Pj)
            {
                  if ($Pdato eq "verdadero")
                  { #Como es verdadero no encontró letras por lo tanto no
se
                  #imprime en el archivo
                        last;
```

```
                    }
                    for (my $Pi=$minx;$Pi<=$maxx;++$Pi)
                    {
                          print LETRAS "$ventana[$Pi][$Pj]";
                    }
                    print LETRAS "\n";
              }
              print LETRAS "letra\n";
#print "Quizás se imprimió una letra\n";
#die "terminación temporal";

#FIN DE LINEAS DE IMPRESION DE LA VENTANA

#Hasta aquí salta cuando los cuadros de inf son muy pequeños
ESCAPE:
      }#Fin del super While
#LINEAS DE PRUEBA DE IMPRESION DE LA VENTANA
#Solo cierra el archivo donde guardo temporalmente las letras
      close(LETRAS);
#FIN DE LINEAS DE PRUEBA DE IMPRESION DE LA VENTANA

# Al final regresa el número de letras que ha encontrado así como el
número de líneas, en el
# número de letras no se incluye el cambio de línea, ni el espacio en la
cuenta
return($letras,$lineas);
} #Fin de recortar
#-------------------------------------------------------------------------
----------------------
sub procesar
{
$tx=$main::tx;
$ty=$main::ty;
$tolx=$main::tolx;
$toly=$main::toly;
#print "lo que recibe recortar $tx $ty $tolx $toly\n";
$IenBlanco=&CreaMatriz;
if ($IenBlanco eq "falso")
{
      ($letras,$lineas)=&recortar($Primerx,$Primery,$Ultimox,$Ultimoy,$tx
,$ty,$tolx,$toly);
      #Cargo el archivo letras.txt en la ventana del programa principal
      #$main::txtPrincipal->Load("letras.txt");
# Destruyo GranMatriz y a su gemelo para liberar memoria
@GranMatriz=();
@GemeloGranMatriz=();
# NUEVO llamo a la función de identificación si entreno es "no"
if($main::entreno eq "no")
{
#Llamo a la función del identificador
&identificador::qLetraEs($main::usuario);
print "usuario que se esta usando $main::usuario\n";
}
      #Imprimo el número de letras y líneas que se ha encontrado
      $main::txtSecundaria->insert('end',"Se ha encontrado $letras letras
y $lineas líneas\n");
}
```

```
else
{
      print "¡Imagen en blanco!\n";
      $main::txtSecundaria->insert('end',    "¡La    imagen    esta    en
blanco!\n");
}
} #Fin de procesar
#-------------------------------------------------------------------------
----------------------
sub Gemelo
{ # Función para el entrenamiento y crea una matriz que es gemela a
GranMatriz
      for (my $j=0;$j<=($alto-1);++$j)
      {
            for (my $i=0;$i<=($ancho-1);++$i)
            {
                  $GemeloGranMatriz[$i][$j]=$GranMatriz[$i][$j];
            }
      }


# Lo hace debido a que recortar borra el contenido de GranMatriz, por
tanto necesito una matriz
# de respaldo o tendría que pasar otra vez el proceso de CreaMatriz.
}
#-------------------------------------------------------------------------
----------------------
sub UsarGemelo
{ # Función para el entrenamiento y asigna a GranMatriz su contenido
original guardado en la
# matriz GemeloGranMatriz
      for (my $j=0;$j<=($alto-1);++$j)
      {
            for (my $i=0;$i<=($ancho-1);++$i)
            {
                  $GranMatriz[$i][$j]=$GemeloGranMatriz[$i][$j];
            }
      }

}
#-------------------------------------------------------------------------
----------------------
1; #Valor de retorno, que indica el fin del módulo
```

A.4.6 *art2*

```
# Módulo de la red ART 2 (analógica)
# Modificado 17/2/04
# Autor Yo - M

#Rutinas existentes:
# art2Inicio(m)  inicializa la red con configuraart2.ini, después de
esta rutina se pueden cargar
#          lo pesos desde un archivo externo, m representa el número de
entradas.
```

```
# art2_1    copia el vector de entrada
# art2_2    pasa I a W
# art2_3    pasa W a X
# art2_4    pasa X a V
# art2_5    V pasa a la subcapa U
# art2_6    U pasa a P, con la influencia de sigmaZ
# art2_7    P pasa a Q
# art2_8
# art2_9
# art2_10   Llama a las sigs rutinas y dará el resultado, devuelve un
escalar "$Resultado"
# art2_11
# art2_12
# art2_13
# art2_14
# art2_15
# art2_guardar    Escribe los pesos en un archivo
# art2_leer Lee los valores de peso de un archivo
# mag1D           Valor absoluto de un vector, regresa un escalar
# VentE           Vector entre Escalar, regresa un vector
# VporE           Vector por un Escalar, regresa un vector
# art2_procesa    Después de llamar al primer paso esta función se
encarga de lo demás y de dar
#           una respuesta


#Líneas de prueba, el programa que llame a éste módulo tiene que hacerlo
de la misma manera que
# aquí, pero faltan algunos arreglos
# Se llama a art2Inicio ( se llama solo una vez al inicio)
# Después se pueden leer pesos almacenados y/o mandar la entrada a art2_1
(la llamada de pesos se
# hace una vez al inicio, mandar entradas todas las veces que sea
necesario)
# Se llama a art2_procesa y este da la respuesta (todas las veces que sea
necesario)
# así:
# &art2Inicio;          Inicializa la red con la config de
configuraart2.ini
# &art2_leer("art2p.txt");    Si es que se desea cargar los pesos de un
archivo
# @entrada=(1, 1, 1, 1, 1);   Datos de entrada(Ent>=0)
# art2_1(@entrada);           Llama al primer paso y le manda los datos
# $respuesta=&art2_procesa;   Se obtiene el número de nodo o mensajes de
no clasificado
# &art2_guardar("art2p.txt"); Para escribir los pesos encontrados

package art2;

#_____________________________________________________________________
_____________________
#Rutinas

#*****************
sub art2Inicio
{
# El número de entradas es pasado como argumento
```

```
      $M=$_[0];
#Se leen los valores del archivo de configuración configuraart2.ini

      open(CONFIGURACION,"configuraart2.ini") or die "¡No se puede abrir
el archivo de configuración configuraart2.ini!";
      chomp($a=<CONFIGURACION>);
      chomp($b=<CONFIGURACION>);
      chomp($c=<CONFIGURACION>);
      chomp($d=<CONFIGURACION>);
      chomp($e=<CONFIGURACION>);
      chomp($theta=<CONFIGURACION>);
      chomp($rho=<CONFIGURACION>);
      chomp($N=<CONFIGURACION>);    #número de salidas

      for ($i=0;$i<=($M-1);++$i)
      {
            for ($j=0;$j<=($N-1);++$j)
            {
                  $Zij[$i][$j]=0;                     #Pesos  descendentes
= 0
                  $Zji[$j][$i]=0.9/((1-$d)*sqrt($M)); #Pesos ascendentes
            }


      }

      close(CONFIGURACION);
# Variable de prueba para que no llame ganadora a una misma neurona en la
fase de entrenamiento
# la llamada de cargar pesos desde un archivo es la única manera de
establecer este valor a 'no'
      $entreno='si';

      @GanAnt=();       #  NUEVO. esta variable llevará el conteo de
neuronas ganadoras en
                        # esta fase de entreno para no escoger la misma
salida para dos
                        # datos que se tiene seguridad son distintos
} #Fin de art2Inicio

#-----------------------------

sub art2_1
{     #Necesita un vector de entrada para poder copiarlo a I
      #inicializa algunas variables
      @I=@_;
      #print "recibido @I\n";
      $contador=1;
      @NodosFuera=();         #aquí se pondrán las posiciones de los nodos
fuera de competencia
                        #las posiciones que tengan un uno estarán fuera de
competencia
      $NG=-10;          #La  neurona  ganadora  no  existe  al  principio,
además este valor
                        #negativo me servirá con el sigmaZiJ y para marcar
nodos
                        #desactivados en la rutina 10
```

```
} #Fin de art2_1

#------------------------------

sub art2_2
{     #Se pasa I a W

      @W=();
      @X=();
      @V=();
      @U=();
      @P=();                    #Inicialización de vectores...
      @Q=();
      @R=();
      @T=();
      @G=();
      @sigmaZiJ=();
      for ($i=0;$i<=($M-1);++$i)
      {
            $W[$i]=$I[$i]+$a*$U[$i];
      }

#print "vector W @W\n U @U\n";
} #Fin de art2_2

#------------------------------

sub art2_3
{     #W va a X
      $magW=&mag1D(@W); #llamo a la función de magnitud de vector
      $Esc=$e+$magW;
      @X=&VentE(@W,$Esc);     #llamo a la función de vector entre escalar
} #Fin de art2_3

#------------------------------

sub art2_4
{     #X se propaga a la capa V, V=f(Xi)+bf(Qi), f=0 para X,Q<=theta
      for ($i=0;$i<=($M-1);++$i)
      {

      $V[$i]=($X[$i]<=$theta?0:$X[$i])+$b*($Q[$i]<=$theta?0:$Q[$i]);
      }
#     print "capa V @V \n";
} #Fin de art2_4

#------------------------------

sub art2_5
{     #V se propaga a la subcapa U
      $magV=&mag1D(@V); #llamo a la función de magnitud de vector
      $Esc=$e+$magV;
      @U=&VentE(@V,$Esc);     #llamo a la función de vector entre escalar
} #Fin de art2_5

#------------------------------
```

```
sub art2_6
{ #se propaga U hacia P
      @sigmaZiJ=();
      if ($NG!=(-10))
      { #Si existe un ganador se multiplica cada valor de G por la
primera fila de Zij,
      # se suman los productos y se obtiene el primer valor de sigmaZiJ,
y se continúa con la
      # sig. fila de Zij, el -10 lo pongo en la parte 2

            for($i=0;$i<=($M-1);++$i)
            {     $temp=0;
                  for($j=0;$j<=($N-1);++$j)
                  {
                        $temp+=$G[$j]*$Zij[$i][$j];
                  }
                  $sigmaZiJ[$i]=$temp;
            }
      }
      for ($i=0;$i<=($M-1);++$i)
      {

            $P[$i]=$U[$i]+$sigmaZiJ[$i];
      }
} #Fin de art2_6

#------------------------------

sub art2_7
{ #P se propaga a Q
      $magP=&magnD(@P); #llamo a la función de magnitud de vector
      $Esc=$e+$magP;
      @Q=&VentE(@P,$Esc);     #llamo a la función de vector entre escalar
} #Fin de art2_7

#------------------------------

sub art2_8
{ #repito los pasos del 2 al 7 para estabilizar los valores
      &art2_2;
      &art2_3;
      &art2_4;
      &art2_5;
      &art2_6;
      &art2_7;
      &art2_2;
      &art2_3;
      &art2_4;
      &art2_5;
      &art2_6;
      &art2_7;

} #Fin de art2_8

#------------------------------

sub art2_9
```

```
{ #Se calcula la salida de la capa R

      $magP=&mag1D(@P);
      $magU=&mag1D(@U);
      $Esc=$e+$magU+$c*$magP;
      @divU=&VentE(@U,$Esc);
      @divP=&VentE(@P,$Esc); #Lo divido en dos partes debido a una
multiplicación que hay que
                        #hacer para el vector P
      for ($i=0;$i<=($M-1);++$i)
      {
            $R[$i]=$divU[$i]+$c*$divP[$i];
      }
} #Fin de art2_9

#------------------------------

sub art2_10
{  #Determina si hay restauración y llama a las rutinas que sean
apropiadas, si hay restauración
# marca el nodo como desactivado,
      $magR=&mag1D(@R);
      $resonancia=($rho/($e+$magR))>1?"no":"si";      #si no se cumple la
ec hay resonancia
      if ($resonancia eq "no")
      {     #Si no hay resonancia se marca el nodo como desactivado y se
comienza desde la
            #rutina 2
            $Resultado="Restauración";
            if($NG!=-10)
            { #Indica que ya se escogió un nodo ganador, por lo tanto se
marca ese nodo como
            #desactivado
                  $NodosFuera[$NG]=1;
                  print "Nodo $NG fuera\n";
            }
            else
            {
            # Este caso ocurre cuando no se puede clasificar una entrada
en la primera vuelta
            # pues $rho/($e+$magR) es mayor que 1 aunque sea por unas
pocas décimas, lo cual
            # puede ser producido por el redondeo
                  return("Todos Fuera");
            }
            @W=();
            @X=();
            @V=();
            @U=();
            @P=();                  #Inicialización de vectores...
            @Q=();
            @R=();
            @T=();
            @G=();
            #Temporalmente he puesto que vuelva a 1 el contador ¡¡HAY QUE
REVISARLO!!
            $NG=-10;
```

```
            $contador=1;
            return($Resultado);
      }
      elsif ($contador==1)
      { #si es 1 significa que parece estar bien clasificado pero falta
hacer la segunda
      #vuelta para asegurarse
            ++$contador;      #incrementa el contador
            &art2_11;
            $Resultado=&art2_12;
            if ($Resultado eq "Todos Fuera")
            { #Termina esta rutina pues no hay nodos que compitan
                  return($Resultado);
            }
            #La idea es que la rutina 13 llamará otra vez a la rutina 10,
y otras más, por
            #tanto el resultado final se regresará en la rutina 13, por
tanto aquí termina
            #el proceso, puede regresar Todos Fuera, Restauración o un
valor de $NG
            $Resultado=&art2_13;
            return($Resultado);
      }
      elsif ($contador==2)
      { #si el contador es igual a 2 indica que si hay resonancia y ya se
clasifico
      #correctamente
            &art2_14;
            &art2_15
      }

} #Fin de art2_10

#------------------------------

sub art2_11
{ #Se propaga P hasta F2
      for ($j=0;$j<=($N-1);++$j)
      {
            $temp=0;
            for ($i=0;$i<=($M-1);++$i)
            {
                  $temp+=$P[$i]*$Zji[$j][$i];
            }
            $T[$j]=$temp;
      }

#     print "T = @T\n";
} #Fin de art2_11

#------------------------------

sub art2_12
{ #Se selecciona el nodo ganador o manda el mensaje de que todos los
nodos están fuera de
#competencia
      for ($j=0;$j<=($N-1);++$j)
```

```
      { #Este for es para encontrar el primer nodo válido para competir
            if   ($NodosFuera[$j]==0   and   !(($entreno   eq   'si')   and
($GanAnt[$j]==1)))
            { #El nodo puede competir
                  $NG=$j;           #Se marca como primer posible ganador
                  last;       #se termina el for pues ya hay ganador
            }
            else
            {
                  if ($GanAnt[$j]==1)
                  {
                        print "se quiso agarrar $j otra vez\n";
                  }
                  $NG=-10;    #Indica que todavía no hay nodo ganador
            }
      }

      if ($NG==-10)
      { #Todos los nodos están fuera de competencia
            return("Todos Fuera");
      }

      for ($j=$NG+1;$j<=($N-1);++$j)
      {
      #Si hay nodo ganador entonces empiezo a buscar el mayor valor en T
            if   ($NodosFuera[$j]==0   and   !(($entreno   eq   'si')   and
($GanAnt[$j]==1)))
            { #Comparo T anterior con T presente para encontrar el mayor
                  $NG=$T[$NG]<$T[$j]?$j:$NG;
                  #En NG esta el número de nodo ganador
            }
      }

      for ($j=0;$j<=($N-1);++$j)
      { #Se crea G, pone $d en el nodo ganador, 0 en caso contrario
            $G[$j]=$NG==$j?$d:0;
      }

      #Regresa la posición del nodo ganador
      return($NG);
} #Fin de art2_12

#------------------------------

sub art2_13
{ #Regresa el valor de la rutina10 que ella llama a la rutina 10 de la
que fue llamada
# que es quien toma la decisión de restaurar, regresar el ganador o todos
fuera
      &art2_6;
      &art2_7;
      &art2_8;
      &art2_9;
      $rutina10=&art2_10;
      return($rutina10);
} #Fin de art2_13
```

```
#------------------------------

sub art2_14
{ #Se modifican los pesos ascendentes de la unidad ganadora F2
      print "Neurona Ganadora NG $NG\n";
#NUEVO marco la pos. de la ganadora para que no se ocupe de nuevo durante
el entrenamiento
$GanAnt[$NG]=1;
#NUEVO fin
      for ($i=0;$i<=($M-1);++$i)
      { #ZJi=Ui/(1-$d)
            $Zji[$NG][$i]=$U[$i]/(1-$d);
      }
} #Fin de art2_14

#------------------------------

sub art2_15
{ #Se modifican los pesos descendentes de la unidad ganadora
      for ($i=0;$i<=($M-1);++$i)
      {
            $Zij[$i][$NG]=$U[$i]/(1-$d);
      }
} #Fin de art2_15
#------------------------------
sub art2_guardar
{ #Graba los pesos en el archivo que le es mandado como argumento
      my $destino=$_[0];
      open(DESTINO,">$destino") or die("No se pudo abrir el archivo
especificado para escribir los pesos");
      print "nombre del archivo $destino";

      for ($j=0;$j<=($N-1);$j++)
      {
            for ($i=0;$i<=($M-1);$i++)
            {
                  print DESTINO "$Zji[$j][$i] ";
            }
            print DESTINO "\n";

      }

      for ($i=0;$i<=($M-1);$i++)
      {
            for ($j=0;$j<=($N-1);$j++)
            {
                  print DESTINO "$Zij[$i][$j] ";
            }
            print  DESTINO "\n";

      }
      close(DESTINO);
} #Fin de art2_guardar
#------------------------------
sub art2_leer
{ #Lee los pesos del archivo que se le halla especificado como argumento
      my $origen=$_[0];
```

```
      print "nombre del archivo $origen";

# Para indicar que no se esta realizando un entreno sino un
reconocimiento
      $entreno='no';
      print " entreno = $entreno entradas $M\n";

      open(ORIGEN,"< $origen" ) or die("No se pudo abrir el archivo
especificado para leer los pesos");
      for ($j=0;$j<=($N-1);$j++)
      {
            $temporal=<ORIGEN>;     #Se almacena una línea de datos tengo
que partirlo
            chop($temporal);
            @temporal=split(/ /,$temporal);
            for ($i=0;$i<=($M-1);$i++)
            {
                  $Zji[$j][$i]=$temporal[$i];
            }
      }
      for ($i=0;$i<=($M-1);$i++)
      {
            $temporal=<ORIGEN>;     #Se almacena una línea de datos tengo
que partirlo
            chop($temporal);
            @temporal=split(/ /,$temporal);
            for ($j=0;$j<=($N-1);$j++)
            {
                  $Zij[$i][$j]=$temporal[$j];
            }
      }
      close(ORIGEN);
print "se han cargado desde $origen\n";

} #Fin de art2_leer
#------------------------------
sub art2_procesa
{
      $Fin="Restauración";
      while($Fin eq "Restauración")
      { #repite el proceso mientras ocurra restauración y aún queden
nodos compitiendo
      &art2_2;
      &art2_3;
      &art2_4;
      &art2_5;
      &art2_6;
      &art2_7;
      &art2_8;
      &art2_9;
      $Fin=&art2_10;
      }

      if ($Fin eq "Todos Fuera")
      { #Cuando todos los nodos quedan fuera de competencia entonces
termina el proceso
      # y regresa a quien llamo la rutina el valor No clasificable
```

```
            return("No clasificable");
      }
      else
      { #caso contrario ha sido clasificado y regresa el número de nodo
que lo
      # identifica para poder seleccionar la letra relacionada con ese
nodo
            return($NG);
      }
} #Fin de art2_procesa
#------------------------------
#_____________________________________________________________________
_____________________
#Otras funciones que ocupo en el proceso

sub mag1D
{     #Encuentra la raíz cuadrada de la suma de los cuadrados de los
elementos del vector de
      #entrada, devuelve un valor escalar
      my @entrada;
      my $i;
      my $resultado;
      my $SumCuad;
      my $largo;
      $SumCuad=0;
      @entrada=@_;            #copio el vector de entrada para modificarlo
sin problemas
      $largo=@entrada;  #averiguo el tamaño del vector de entrada
      for ($i=0;$i<=($largo-1);++$i)
      {
            $SumCuad+=($entrada[$i])**2;
      }
      $resultado=sqrt($SumCuad);
      return($resultado);
}     #Fin de mag1D

#------------------------------

sub VentE
{
      my $ultimo;
      my @entrada;
      my @resultado;
      my $largo;
      my $i;
      @entrada=@_;
      $ultimo=pop(@entrada);        #el último elemento es el escalar, por
eso lo tomo
      $largo=@entrada;        #tamaño del vector de entrada
      for ($i=0;$i<=($largo-1);++$i)
      {
            $resultado[$i]=$entrada[$i]/$ultimo;
      }

      return(@resultado);
}
```

```
#------------------------------

1;    #Fin de todo
```

A.4.7 *zoom*

```
# Este módulo sirve para implementar un zoom que puede tener un factor
para x
# y otro factor distinto para y, esto con el fin de aprovechar por
completo el cuadro de
# segmentación y mejorar con ello el reconocimiento, se puede eliminar la
parte de regularizar
# el tamaño que se hace en la parte de la tdc2
# NOTA: Las letras que sean mucho menores en ancho o alto al tamaño de tx
o ty no será variado
# en la dimensión en que es demasiado pequeña, esto para no confundir .
con - o i con alguna otra
# cosa.
# Revisada: 17/2/04

# Posee una única función llamada zoom, no necesita argumentos
package zoom;

#------------------------------------------------------------------------
-----------------------
sub zoom
{
open(ORIGEN,'letras.txt');
open(DESTINO,'>letras2.txt');
my @MatLinea=();        # Matriz donde se guarda la línea leída desde
letras.txt
my @MatLetra=();        # Matriz donde se guardan las líneas de una letra
my $F=0;                # Cantidad de filas leídas
my $C=0;                # Cantidad de columnas leídas
my $CuentaPix=0;        # Cantidad de pixeles leídos
my $limF=0;             # Límite en el número de filas del lazo para
interpolación
my $limC=0;             # Límite en el número de columnas del lazo para
interpolación
my $escC=0;             # Valor de la escala en C, col
my $escF=0;             # Valor de la escala en F, filas
my $filaE=0;                 # Fila escogida para tomar de MatLetra a
MatLetra2
my $colE=0;             # Col escogida para tomar de MatLetra a MatLetra2
my $temp=0;             # Variable de usos varios y temporales
# Averiguo si la variable main entreno es si o no
$ty=$main::entreno eq 'si'?$nusuario::tyini:$main::tyini;
$tx=$main::entreno eq 'si'?$nusuario::txini:$main::txini;
while($hilera=<ORIGEN>)
{
      chomp($hilera);
# En caso de ser espacio o Intro o ""
      if ($hilera eq 'espacio' or $hilera eq 'Intro' or $hilera eq '')
      {
            if($hilera ne '')
```

```
            {
            #print "$hilera\n";
            print DESTINO "$hilera\n";
            }
      }
      elsif($hilera eq 'letra')
      {

#*****************************************************
# Se toma cada dato y se pasa el aumento de la imagen, de ser necesario,
la fuente es
# MatLetra y el destino MatLetra2
$C=int($CuentaPix/$F)+0;
#print "Filas $F Col $C total $CuentaPix\n";
# Siempre genera MatLetra2 con ceros que después serán rellenados por
otros valores
for (my $f=0;$f<=($ty-1);++$f)
{
      for (my $c=0;$c<=($tx-1);++$c)
      {
            $MatLetra2[$f][$c]=0;
      }


}

# Ahora establezco los límites y la escala dependiendo de la rel entre
tx, tx, F y C
if ($C<($tx/3))
{
# Si la cantidad de columnas encontradas es menor que un tercio de tx, no
quiero que se aumente
# en la dir x, solo quiero que se copien los datos por tanto escala=1 y
limite =C
      $escC=1;    # No hay aumento en la dir x
      $limC=$C;   # Solo copia los datos a sus posiciones originales
}
else
{
# La cantidad es mayor por tanto quiero afectar su apariencia. La escala
será tx/C y el limite en
# la dir x será tx
      $escC=$tx/$C;     # Tantas veces como sea menor C
      $limC=$tx;  # Afecta todo MatLetra2
}
if ($F<($ty/5))
{
# Si la cantidad de filas encontradas es menor que un quinto de tx, no
quiero que se aumente
# en la dir y, solo quiero que se copien los datos por tanto escala=1 y
limite =F
      $escF=1;    # No hay aumento en la dir y
      $limF=$F;   # Solo copia los datos a sus posiciones originales
}
else
{
```

```
# La cantidad es mayor por tanto quiero afectar su apariencia. La escala
será ty/F y el limite en
# la dir y será ty
      $escF=$ty/$F;     # Tantas veces como sea menor F
      $limF=$ty;  # Afecta todo MatLetra2
}

# LAZO QUE EFECTUA LOS CAMBIOS DE APARIENCIA:
for (my $f=0;$f<=($limF-1);++$f)
{
      for (my $c=0;$c<=($limC-1);++$c)
      {
      # Obtengo la pos de MatLetra de donde se obtendrá el valor
      # El redondeo es necesario para tener valores enteros aproximados
de donde se obtendran
      # los datos, ya que las posiciones son valores enteros
            $temp=$f/$escF;
            $filaE=int(sprintf("%.0f",$temp));
            $filaE=$filaE>=$F?$filaE-1:$filaE;
            $temp=$c/$escC;
            $colE=int(sprintf("%.0f",$temp));
            $colE=$colE>=$C?$colE-1:$colE;
            $MatLetra2[$f][$c]=$MatLetra[$filaE][$colE];
      }


} # Fin del lazo para cambiar la apariencia de la letra

# Lazo para imprimir en el archivo letras2.txt lo que se encuentre en
MatLetra2
for (my $f=0;$f<=($ty-1);++$f)
{
      for (my $c=0;$c<=($tx-1);++$c)
      {
            print DESTINO "$MatLetra2[$f][$c]";
      }
      print DESTINO "\n";
}
print DESTINO "letra\n";
@MatLinea=();
@MatLetra=();
$F=0;
$C=0;
$CuentaPix=0;
#*****************************************************
      }
      else
      {
      #
            ++$F;
            @MatLinea=split(//,$hilera);
            push @MatLetra,[@MatLinea];
            $CuentaPix+=(@MatLinea+0);
      }

} # Fin del while
sleep(2);   # retardo para darle tiempo a escribir los datos
```

```
close(DESTINO);
} # Fin de zoom
1; # Para validar el módulo
```

A.4.8 *vecino8*

```
# Función para sacar la tdc 2.
# Entrada: el nombre de un archivo que este compuesto de 1 y 0, además
ignora las entradas
# "Intro", "espacio", "letra", junto con el nombre debe ir la cantidad de
valores que deberán
# ser devueltos por la función
# Salida: un archivo con cadenas de valores donde antes habían letras,
dichas cadenas de valores
# deberán contener tantos elementos como entradas tenga la red.

# Revisado el: 17/02/04
# Varié la terminación de los lazos de procesamiento para que solo tomen
hasta tx - 1 y ty-1
# con el objeto de regularizar el tamaño y forma de la ventana a procesar
# Crea una serie de datos que son simplemente la sumatoria de los pixeles
en su vecindario ocho
# más su propio valor, si es negro se establece en 1/9 y se le suma la
cantidad de vecinos
# 1 vecino=1/9, 2 vecinos=2/9 etc.
package vecino8;

sub tdcm
{
      system("cls");
      print "Se esta ejecutando el vecino8...\n";

      my $Darchivo;           # dirección del archivo a transformar
      my $Nvalores;           # cantidad de valores a ser escritos por
letra en el DESTINO
      my $hilera;       # Línea leída desde ORIGEN
      my $CuentaPixeles;      # Cuenta los pixeles que hay para una letra
      my @MatLinea;           # Matriz donde se almacena el contenido de
una línea de datos
      my @MatLetra;           # Matriz que almacena todos los pixeles de
una letra (2-D)
      my $M;                  # Número de filas de @MatLetra
      my $N;                  # Número de columnas de @MatLetra
# Se obtiene el nombre del archivo con el que se va a trabajar y el
número de valores que sacará
# por cada letra.

      $Darchivo=$_[0];
      $Nvalores=$_[1];
      print "archivo = $Darchivo, n = $Nvalores\n";
      open(ORIGEN,"$Darchivo") or die "No se pudo abrir el archivo
$Darchivo";
      open(DESTINO,">destino.txt") or die "No se pudo abrir el archivo
destino";
      open(RE,">replica.txt") or die "No se abrio replica";
```

```
      # Averiguo si la variable main entreno es si o no
      $ty=$main::entreno eq 'si'?$nusuario::tyini:$main::tyini;
      $tx=$main::entreno eq 'si'?$nusuario::txini:$main::txini;
# Reinicio los contadores de pixeles, de líneas y columnas
      $CuentaPixeles=0;
      $M=0;
      $N=0;
# Este while se encarga de leer el archivo hasta el final

      while($hilera=<ORIGEN>)
      {
      # Evalúa los posibles valores que puede tener la hilera, para Intro
y espacio, no cambia
      # los valores y los escribe en el archivo de salida.
            chomp($hilera);
            if (($hilera eq "Intro") or ($hilera eq "espacio"))
            {
                  #print "$hilera \n";
                  print DESTINO "$hilera\n";
            }
            elsif ($hilera eq "letra")
            {
            # Para este caso debe finalizar la búsqueda de inf para
realizar la tdc para esa
            # letra, pero si la cantidad de datos encontrada es menor que
el n pedido se pasa
            # a realizar la sig búsqueda.
REALIZATDC:
# Salto cuando hay muy pocos pixeles y se agregan columnas para llegar al
N pedido
                  $M=int($M);
                  $N=int($CuentaPixeles/$M);
                  print "filas $M col $N acumulado $CuentaPixeles tx $tx
ty $ty\n";
                  if(($CuentaPixeles >= $Nvalores) and ($N>=$tx) and
($M>=$ty))
                  {

#-------------------------------------------------------------------------
-----------------------


# Se pasó al llamado de la función TDC2 la cantidad de datos que se
necesitan, por tanto va
# a almacenar solamante la cantidad de datos solicitados,
# matriz donde se encuentran los coeficientes y procederá a imprimirlos
en el archivo DESTINO
# Por cada vecino con color se aumenta su valor en 1/8

for (my $j=0;$j<=($ty-1);++$j)
{
      for (my $i=0;$i<=($tx-1);++$i)
      {
            $dato=0;    # Se inicializa en cero y se investiga cada vecino
      # Establezco los limites inferiores para evitar la pos. -1, las
posiciones superiores
      # no importan ya que los pone como ''
```

```
            my $fl=($j-1)>=0?($j-1):$j;
            for (my $f=$fl;$f<=($j+1);++$f)
            {
            # Establezco la columna inferior
                  my $cl=($i-1)>=0?($i-1):$i;
                  for (my $c=$cl;$c<=($i+1);++$c)
                  {
                        $dato+=$MatLetra[$f][$c]*1/9;
                  }

            }
            print DESTINO "$dato ";
            print RE "$dato";
if ($MatLetra[$j][$i] eq "")
{
print "Malo no hay fila $j col $i";
}
      }
      print RE "\n";

}
print DESTINO "\n";
print RE "\n";

#-------------------------------------------------------------------------
----------------------



                  }     #Fin del then para realizar la TDC o la no tdc
                  else
                  {
                  # Agrega ceros a las columnas hasta obtener el número
de datos solicitado
                  # después salta
                        print "Muy pocos datos, se agregan ceros\n";
                  # Encuentro el número de columnas
                        $CuentaPixeles=int($CuentaPixeles);
                        $M=int($M);
                        $tcol=int($CuentaPixeles/$M);
                        $tcol=int($tcol);
                        $N=$tcol;
                  # Agrego filas de ceros hasta superar la cuenta de ty
pedido
                        --$M;
                        --$M;
                        my $j=0;
                        while($M <($ty-1))
                        {
                              ++$M;
                        # Realiza este lazo hasta superar o igualar a ty
                              for ($agrega=0;$agrega<$tcol;++$agrega)
                              {
                              #  Replico lo que aparece en la parte
superior de la
                              # letra para rellenar el resto de la imagen
```

```
      #$MatLetra[$M][$agrega]=$MatLetra[$j][$agrega]*1;
                                    $MatLetra[$M][$agrega]=0;
                              }
                              ++$j;
                              $CuentaPixeles+=$tcol;
                        }
                        ++$M;
                        --$N;
                        --$N;
                        my $i=0;
                        while ($N<($tx-1))
                        {
                              ++$N;
                              for ($agrega=0;$agrega<$M;++$agrega)
                              {
                                    $MatLetra[$agrega][$N]=0;
                              #  Replico lo que aparece en la parte
superior de la
                              # letra para rellenar el resto de la imagen

      #$MatLetra[$agrega][$N]=$MatLetra[$agrega][$i]*1;
                              }
                              ++$i;
                              $CuentaPixeles+=$M;
                        }
                        ++$N;
                        print "Quedo con $M filas y $N columnas\n";

                        goto REALIZATDC;

                  }     #Fin del else para realizar el vecino8

            # Al realizarse la función tdc reinicia los contadores para
pasar a la sig letra
                  $CuentaPixeles=0;
                  $M=0;
                  $N=0;
                  @MatLinea=();
                  @MatLetra=();
            }
            else
            {
            # Lo que se ha encontrado es una cadena de inf por tanto
divide la hilera y la
            # asigna a una matriz, además aumenta la cuenta de datos para
saber si cumple
            # con el n pedido.
                  #print "cadena $hilera encontrada\n";
                  ++$M;
                  #print "hilera $M ";
                  @MatLinea=split(//,$hilera);
                  push @MatLetra,[@MatLinea];
                  $CuentaPixeles+=(@MatLinea+0);
                  $CuentaPixeles=int($CuentaPixeles+0);
```

```
            }
      }
close DESTINO;
print "Se han guardado todos los datos para el vecino8\n";
}
1;
```

A.4.9 *identificador*

```
# Módulo para llamar a la red ART2 mandarle los datos que se encuentra en
el archivo destino.txt
# e imprimir el caracter identificado en la pantalla.
# Este módulo es usado después de la segmentación regresa un conjunto de
caracteres que son
# impresos en la ventana de texto principal, le pasa el enfoque a esta y
destruye la ventana del
# segmentador
# Revisada 17/02/04
# Funciones que se encuentran en este módulo:
# &qLetraEs("usuario") abre el archivo de relacion y pesos del usuario,
llama a la art2 e
#                identifica los caracteres para imprimirlos en pantalla

# pero los he puesto como comentarios para probar otra idea, excepto por
el zoom, ese sigue
# funcionando

package identificador;
sub qLetraEs
{
$usuario=$_[0];
# Muestra el resultado de la segmentación para que el usuario pueda
variar los parámetros del
# segmentador y realizar otra iteración hasta que le parezca la
segmentación
# Rutina usada para mostrar el resultado de una segmentación
      $ventanota=$main::ventanota->Toplevel();
      $lblTexto=$ventanota->Label(-text=>"Se han segmentado
$segmentar::letras letras y $segmentar::lineas líneas.\nSi el resultado
de la segmentación es satisfactorio presione el botón 'Acepto el
resultado', de lo contrario varie los parámetros del segmentador y
presione 'Realizar otra iteración'. Cualquier consulta acerca de los
parámetros del segmentador puede hacerse en la ayuda del programa.",-
wraplength=>500)->pack(-anchor=>'n');
      $ventanota->configure(-title=>'Resultado de la segmentación');
# Aquí irían los cuadros de opción o entradas para variar los parámetros
de segmentación
      $frmParametros=$ventanota->Frame()->pack(-anchor=>'n');
      $lblTx=$frmParametros->Label(-text=>"Tx")->grid(-column=>0,-
row=>1);
      $entTx=$frmParametros->Entry()->grid(-column=>1,-row=>1);
      $btnTxd=$frmParametros->Button(-text=>'-',-command=>sub{&Txd})-
>grid(-column=>2,-row=>1);
      $btnTxa=$frmParametros->Button(-text=>'+',-command=>sub{&Txa})-
>grid(-column=>3,-row=>1);
```

```
      $lblTy=$frmParametros->Label(-text=>"Ty")->grid(-column=>0,-
row=>2);
      $entTy=$frmParametros->Entry()->grid(-column=>1,-row=>2);
      $btnTyd=$frmParametros->Button(-text=>'-',-command=>sub{&Tyd})-
>grid(-column=>2,-row=>2);
      $btnTya=$frmParametros->Button(-text=>'+',-command=>sub{&Tya})-
>grid(-column=>3,-row=>2);
      $lblTolx=$frmParametros->Label(-text=>"Tol    x")->grid(-column=>0,-
row=>3);
      $entTolx=$frmParametros->Entry()->grid(-column=>1,-row=>3);
      $btnTolxd=$frmParametros->Button(-text=>'-',-command=>sub{&Tolxd})
            ->grid(-column=>2,-row=>3);
      $btnTolxa=$frmParametros->Button(-text=>'+',-command=>sub{&Tolxa})
            ->grid(-column=>3,-row=>3);
      $lblToly=$frmParametros->Label(-text=>"Tol    y")->grid(-column=>0,-
row=>4);
      $entToly=$frmParametros->Entry()->grid(-column=>1,-row=>4);
      $btnTolyd=$frmParametros->Button(-text=>'-',-command=>sub{&Tolyd})
            ->grid(-column=>2,-row=>4);
      $btnTolya=$frmParametros->Button(-text=>'+',-command=>sub{&Tolya})
            ->grid(-column=>3,-row=>4);
# Obtengo los valores de los parámetros
      $tx=$main::tx;
      $ty=$main::ty;
      $tolx=$main::tolx;
      $toly=$main::toly;
      $entTx->insert("0.0","$tx");
      $entTy->insert("0.0","$ty");
      $entTolx->insert("0.0","$tolx");
      $entToly->insert("0.0","$toly");

# Frame para los botones aceptar el resultado, ayuda etc.
$frmBot=$ventanota->Frame()->pack();
$btnC=$frmBot->Button(-text=>'Cancelar el proceso',-command=>sub{
      $ventanota->destroy;
      #$segmentar::ventanota->destroy();
      #$vntDialogo->focus;
      $segmentar::ventanota->focus;
      })->grid(-column=>0,-row=>0);
# Este boton pide otra iteración de la imagen
$btnOtraIteracion=$frmBot->Button(-text=>'Realizar    otra    iteración',-
command=>sub{
      $tx=$entTx->get;  #Obtiene los datos de Tx
      $tx=($tx<1)?1:$tx;
      $tx=int($tx);
      $ty=$entTy->get;  # datos de ty
      $ty=($ty<1)?1:$ty;
      $ty=int($ty);
      $tolx=$entTolx->get;    # datos de tolx
      $tolx=($tolx<1)?1:$tolx;
      $tolx=int($tolx);
      $toly=$entToly->get;    # datos de toly
      $toly=($toly<1)?1:$toly;
      $toly=int($toly);
      $ventanota->destroy;
      &segmentar::procesar;})->grid(-column=>1,-row=>0);
```

```
# Se acepta el resultado además si la segmentación fue correcta aparece
habilitado
      $btnS=$frmBot->Button(-text=>'Acepto el resultado',-command=>sub{
      $ventanota->destroy;
      &qLetraEs2($usuario);})->grid(-column=>2,-row=>0);
# Le paso el enfoque al botón otra iteración
$btnOtraIteracion->focus;
#Llama a la ayuda en main
$btnAyuda=$frmBot->Button(-text=>'Ayuda',-command=>sub{&main::ayudaP;
&ayuda::Tema(3);})
      ->grid(-column=>3,-row=>0);
# Creo frame porque posee scroll

$texto=$ventanota->Scrolled('Pane',-scrollbars=>'osoe')
                  ->pack(-expand=>'yes',-fill=>'both');
# Creo la etiqueta donde aparece la imagen de lo leído del archivo letras
$lblImagen=$texto->Label()->pack();

# Configuro un objeto Photo adentro de la etiqueta para poder dibujar en
ella
$imagen=$lblImagen->Photo();
$lblImagen->configure(-image=>$imagen);

# Lee el archivo letras.txt para mostrar su contenido en una imagen
open(FUENTE,"letras.txt") or die "No se pudo abrir letras.txt";
my $i=0;          # Desplazamiento en x
my $j=0;          # Desplazamiento en y
my @MatLetra=();  # Matriz donde se almacenan los datos de las letras
my @MatLinea=();  # Matriz en que se almacenan las líneas temporalmente
my $M=0;          # Número de lìneas
my $N=0;          # Número de columnas
my $CuentaPixeles=0;    # Cantidad de pixeles encontrados

# Leo línea por línea para obtener los caracteres que forman la imagen de
la letra
while ($hilera=<FUENTE>)
{
      chomp($hilera);
# Lee el dato para variar los parámetros de i y j si es espacio o intro
      if ($hilera eq 'Intro')
      {
      # Es un cambio de línea aumento $j en $ty
            $j+=$ty;
      # Se reinicia $i
            $i=0;
      }
      elsif ($hilera eq 'espacio')
      {
      # Es un espacio, aumento $i en $tx
            $i+=$tx;
      }
      elsif ($hilera eq 'letra')
      {
      # Se ha encontrado el final de una letra por lo que se imprime en
la etiqueta
            $N=$CuentaPixeles/$M;
            for (my $it=0;$it<$M;++$it)
```

```
            {
                  for (my $jt=0;$jt<$N;++$jt)
                  {
                        $imagen->put("$MatLetra[$it][$jt]",-
to=>($jt+$i,$it+$j));
                  }
            }
      # Asigno nuevos valores a las variables de control
            $CuentaPixeles=0;
            $M=0;
            $N=0;
            @MatLinea=();
            @MatLetra=();
            $i+=$tx;

      }
      else
      {
      # Lo que se tiene es una hilera de datos por lo que se agrega a la
matriz
            $hilera=~s/1/black /g;        # Cambio 1 por black
            $hilera=~s/0/white /g;        # Cambio 0 por white
            @MatLinea=split(/ /,$hilera);
            push @MatLetra,[@MatLinea];
            $CuentaPixeles+=@MatLinea;
            ++$M;
      }
}
      $main::txtSecundaria->insert('end',"Se  están  mostrando:  lineas
$lineas letras $letras tolx $tolx toly $toly\n");
            $main::txtSecundaria->see('end');
      print "Muestra dice: tx $tx ty $ty tolx $tolx toly $toly\n";


}

#****************************************

sub qLetraEs2
{
$usuario;         # Variable que contiene el nombre del usuario
my $wUsuario;           # Archivo con los pesos del usuario
my $rUsuario;           # Archivo con la relación del usuario
my $linea;        # Contiene la linea que se esta leyendo
my $insertar;           #  Caracter  a  insertar  en  la  ventana  de  texto
principal
my $archivo;            # Archivo a trabajar con la vecino8
my @MatLinea;           # Matriz que contiene los valores a evaluar en la
art
my $resultado;          # Salida de la art que se activa
my @MatRel;       #   Matriz   donde   se   encuentra   el   contenido   de
$usuario.rel

# Con el nombre del usuario que se le pasa como parámetro carga los
archivos de configuración
# de la red art2 y la relación entre la salida de la red con la letra a
identificar
```

```
      $usuario=$_[0];
      $wUsuario=$main::dir."\\usuarios\\".$usuario.".pes";
      $rUsuario=$main::dir."\\usuarios\\".$usuario.".rel";
      $pUsuario=$main::dir."\\usuarios\\".$usuario.".par";

      open(RELACION,"$rUsuario") or die "no se pudo abrir $rUsuario";
# Paso el contenido del archivo $usuario.rel a una matriz
      @MatRel=<RELACION>;
# Quito los caracteres de fin de línea
      chomp(@MatRel);

# Cantidad de valores que deseo de regreso
      open(PARAMETROS,"$pUsuario") or die "no se pudo abrir $pUsuario";
      @MatPar=<PARAMETROS>;
      chomp(@MatPar);

# Llamo a la función para incrementar el tamaño de los caracteres, el
resultado será evaluado por
# la tdc, crea letra2.txt como salida
      &zoom::zoom;
# Llamo a la función tdc para transformar el archivo "letras2.txt" y me
da como resultado el
# archivo "destino.txt", que contiene los coeficientes para introducir en
la red
      $archivo="letras2.txt";
# Obtengo la cantidad de entradas de la red, que es por tanto el número
de datos que deseo de
# la tdc
      $n=$MatPar[4];
      &vecino8::tdcm($archivo,$n);
#Inicializo la red con el número de entradas que necesito
      &art2::art2Inicio($n);
      print "Red inicializada desde identificador con $n entradas\n";
# Se carga la red con los pesos del usuario
      &art2::art2_leer($wUsuario);
      $art2::entreno="no";
      print "Pesos de $usuario cargados\n";
# Se abre el archivo destino.txt
      open(COEF,"destino.txt") or die "no se pudo abrir destino.txt";
# Inicia el lazo para llamar a la red art2
      while ($linea=<COEF>)
      {
      chomp($linea);
      #Lee una línea de destino.txt si es "Intro" o "espacio" o ""
inserta directamente
      # el caracter en la ventana de texto principal, con "no datos" se
asume que es el "."
            if(($linea eq "Intro") or ($linea eq "espacio") or ($linea eq
""))
            {
                  if($linea eq "Intro")
                  {
                  # En caso de que sea Intro indica un cambio de línea
                        $insertar="\n";
                  }
                  elsif($linea eq "espacio")
                  {
```

```
                    # espacio indica que se ha encontrado un caracter de
espacio
                          $insertar=" ";
                    }
                    else
                    {
                    # De lo contrario es no datos por tanto &
                          $insertar="&";
                    }
                    #print "caracter raro\n";
              }
              else
              {
              # Se tiene una hilera de coeficientes para ser evaluados en
la red
                    @MatLinea=split(/ /,$linea);
                    print "$MatLinea[0] ";
                    &art2::art2_1(@MatLinea);
                    $resultado=&art2::art2_procesa;
              # La neurona de salida que se activa esta relacionada con el
caracter que se
              # encuentra en la pos. de MatRel, por tanto si se activa la
primera neurona
              # indica que el caracter identificado esta en la primera pos.
de la matriz
                    $insertar=$MatRel[$resultado];
              }
        # Se inserta en la ventana principal el caracter reconocido
              $main::txtPrincipal->insert('end',"$insertar");
        } #Fin del while de lectura de lineas
#  Muestra una ventana para preguntar si desea guardar los pesos
resultantes del reconocimiento
# si entreno es no
        if ($main::entreno eq 'no')
        {
        # Presenta la caja de texto para saber si se dese grabar los pesos
resultantes del
        # reconocimiento
              my $dlgGP=$main::ventanota->DialogBox(-title=>'¿Guardar pesos
resultantes?',
                                    -buttons=>['Aceptar','Cancelar']);
              $lblGP=$dlgGP->add('Label')->pack;
              $lblGP->configure(-text=>"¿Desea guardar los pesos
resultantes del reconocimiento de caracteres, se recomienda aceptar si se
ha reconocido correctamente más del 80% de las letras?");
        # Muestra el cuadro de diálogo
              my $btnDialogo=$dlgGP->Show();
              if ($btnDialogo eq 'Aceptar')
              {
              # Llamo a la función para guardar pesos del módulo ART2
                    my
$PesUsuario=$main::dir."\\usuarios\\".$usuario.".pes";
                    &art2::art2_guardar($PesUsuario);
              }

        }
```

```
# Inserta el mensaje de terminado el reconocimiento en la ventana de
texto secundaria
      $main::txtSecundaria->insert('end',"¡Se     ha     terminado     el
reconocimiento!\n");
      $main::txtSecundaria->see('end');
return;
} # Fin de qLetraEs2

#************************************
sub Txd
{
      $tx=$entTx->get;
      --$tx;
      $tx=int($tx);
      $tx=($tx<1)?1:$tx;
      $entTx->delete('0.0','end');
      $entTx->insert('0.0',"$tx");
      $main::tx=$tx;
}

sub Txa
{
      $tx=$entTx->get;
      ++$tx;
      $tx=int($tx);
      $entTx->delete('0.0','end');
      $entTx->insert('0.0',"$tx");
      $main::tx=$tx;
}

sub Tyd
{
      $ty=$entTy->get;
      --$ty;
      $ty=int($ty);
      $ty=($ty<1)?1:$ty;
      $entTy->delete('0.0','end');
      $entTy->insert('0.0',"$ty");
      $main::ty=$ty;
}

sub Tya
{
      $ty=$entTy->get;
      ++$ty;
      $ty=int($ty);
      $entTy->delete('0.0','end');
      $entTy->insert('0.0',"$ty");
      $main::ty=$ty;
}

sub Tolxa
{
      $tolx=$entTolx->get;
      ++$tolx;
      $tolx=int($tolx);
      $entTolx->delete('0.0','end');
```

```
        $entTolx->insert('0.0',"$tolx");
        $main::tolx=$tolx;
}

sub Tolxd
{
        $tolx=$entTolx->get;
        --$tolx;
        $tolx=int($tolx);
        $tolx=($tolx<1)?1:$tolx;
        $entTolx->delete('0.0','end');
        $entTolx->insert('0.0',"$tolx");
        $main::tolx=$tolx;
}

sub Tolya
{
        $toly=$entToly->get;
        ++$toly;
        $toly=int($toly);
        $entToly->delete('0.0','end');
        $entToly->insert('0.0',"$toly");
        $main::toly=$toly;
}

sub Tolyd
{
        $toly=$entToly->get;
        --$toly;
        $toly=int($toly);
        $toly=($toly<1)?1:$toly;
        $entToly->delete('0.0','end');
        $entToly->insert('0.0',"$toly");
        $main::toly=$toly;
}
1;
```