UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

División de Ciencias Básicas e Ingeniería

Proyecto Terminal de Ingeniería en Computación

Sistema de reconocimiento del alfabeto dactilológico utilizando procesamiento de imágenes

Proyecto que presenta:

Marín Díaz Lidia

Para obtener el título de:
Ingeniero en Computación

Asesor:
Dr. Oscar Herrera Alcántara

México, D.F. Julio de 2010

Tabla de contenido

Resumen

En este documento se describe una metodología sobre el reconocimiento de imágenes de letras del alfabeto dactilológico con invariancia a la rotación. El alfabeto dactilológico es un sistema de comunicación que trasmite información mediante el uso de los dedos de la mano.

Se desarrolla un sistema traductor del lenguaje dactilológico que toma como base el procesamiento de imágenes digitales para reconocer letras, con la finalidad de retroalimentar al usuario sobre la correcta postura de la mano y generar un aprendizaje de éste sistema de comunicación.

Existen diversos métodos aplicables al procesamiento de imágenes, un ejemplo de ello es la transformada wavelet discreta. La transformada wavelet discreta permite reducir el tamaño de cada imagen y generar coeficientes de trasformación que representan rasgos característicos de cada letra del alfabeto dactilológico, con lo cual podemos aplicar estos coeficientes a una red neuronal que actúa con memoria asociativa.

La aplicación emplea bibliotecas libres y fue desarrollada en GNU/Linux, con el objetivo de motivar la difusión del lenguaje de señas. Los experimentos validan la eficiencia de este método como sistema de reconocimiento sobre imágenes.

La calidad de las imágenes forman un factor importante al procesar una imagen digital, la iluminación y la tonalidad son factores importantes que enfatizan el éxito o la falla del proceso, para este trabajo se utilizaron imágenes con calidad suficiente, y se obtuvieron resultados aceptables.

Agradecimientos

”Lo que ahoga a alguien no es caerse en el rio,
sino quedarse sumergido en él” P. Coelho

- A mis Padres, gracias totalmente y con todo mi cariño.
- A Dios nuestro señor por permitirme culminar este proyecto.
- A la División de Ciencias Básicas e Ingeniería de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, por la oportunidad de aprender.
- Al Departamento de Electrónica y de Sistemas, por el aprendizaje obtenido.
- Al Dr. Oscar Herrera Alcántara, por su gran apoyo y dirección en el proyecto terminal.
- A Edson Luna por todo su apoyo y cariño brindado. Gracias ...

Capítulo 1
Introducción

El sistema de comunicación dactilológico fijado por la Federación Mundial de Sordos [1], es usado por personas sordo-mudas para comunicarse. La base de este lenguaje es el alfabeto clásico (A, B, C, …, Z) transformado a posturas manuales.

La dactilología es la representación manual de cada una de las letras que componen el alfabeto. A través de ella se puede transmitir a una persona sorda cualquier palabra que se desee comunicar.

Parte fundamental del proyecto es el procesamiento de imágenes, el cual es un campo amplio en el que se cuenta con diversos métodos y algoritmos para lograr una mapeo de información sobre la imagen digital.

En el proyecto se desarrolló una metodología de procesamiento de imágenes aplicada al reconocimiento de letras del alfabeto dactilológico que básicamente consiste en los siguientes pasos.

- Obtener las imágenes de las letras del alfabeto y generar versiones rotadas de las mismas.

- Extraer un conjunto de rasgos característicos que diferencien una imagen de otra empleando la trasformada Wavelet y almacenar los coeficientes de transformación en vectores.

- Entrenar una red neuronal para que reconozca cada uno de los vectores característicos de las letras y sus variantes rotadas.

- Usar la arquitectura de la red neuronal para crear una aplicación software que lea nuevas imágenes que serán reconocidas.

1.1 Motivaciones

En los últimos años se han desarrollado proyectos basados en el reconocimiento de imágenes, en su mayoría, destinados a investigación científica que en pocas ocasiones se traduce a investigación o tecnología aplicada, que lleve un beneficio directo a varios sectores sociales. En México a una gran cantidad de personas sordas les es desconocido o incluso ajeno el alfabeto dactilológico, por lo que es importante crear software que de a conocer ésta forma de comunicación, y así coadyuvar a disminuir esta problemática social.

El uso de las nuevas tecnologías, en particular el de la computadora, ha permitido avances significativos en la comunicación escrita, auditiva y visual. En este sentido, el presente proyecto pretende seguir con esta tendencia y orientarse al reconocimiento del alfabeto dactilológico con la finalidad de ayudar a que los usuarios puedan aprender a usar el alfabeto dactilológico. Considerando que la mayor parte del software para usuarios sordomudos son propietarios y de costo elevado, y esto limita la difusión del conocimiento de este sistema de comunicación, se ha desarrollado el software de reconocimiento con bibliotecas de software libre.

1.2 Objetivos particulares.

Desarrollar un sistema de aprendizaje del alfabeto dactilológico en 2D, basado en reconocimiento de imágenes.

1.3 Objetivos generales.

- Desarrollar un módulo para realizar la segmentación de la imagen.
- Desarrollar un módulo para realizar la extracción de características del objeto que permitan distinguirlo de otro.

- Desarrollar el módulo de presentación de resultados.
- Componer la arquitectura de la aplicación.
- Realizar pruebas.
- Elaborar la documentación

1.4 Organización del reporte

El reporte del proyecto está organizado de la siguiente manera:

- **Capítulo 1. Introducción.** Presenta algunos aspectos como son: la motivación que se tuvo para dirigirla hacia un problema social, el objetivo principal, objetivos particulares, así como el contenido de cada uno de los capítulos que conforman el proyecto.
- **Capítulo 2. Antecedentes**. Presenta los antecedentes que proveen el marco de desarrollo de este proyecto terminal.
- **Capítulo 3. Descripción Técnica**. Presenta los conceptos fundamentales, y las especificaciones técnicas del proyecto terminal.
- **Capítulo 4. Diseño y desarrollo de la aplicación.** Se describen cada uno de los módulos que componen la aplicación software de reconocimiento de imágenes.
- **Capítulo 5. Tecnologías utilizadas**. Presenta las tecnologías utilizadas para desarrollar el sistema de reconocimiento de imágenes.
- **Capítulo 6. Resultados.** Aquí se presentan los resultados experimentales obtenidos al término del proyecto.

CAPÍTULO 1. INTRODUCCIÓN

Capítulo 2
Antecedentes

Ciertamente este proyecto no es pionero en su tipo, existen otros trabajos relacionados que lo anteceden y que se comentan a continuación.

2.1 Referencias externas

”Sistema de reconocimiento de imágenes como intérprete de lenguajes de señas” [2]. Este proyecto utiliza el procesamiento digital de imágenes para realizar traducciones del lenguaje dactilológico; la extracción de características y la definición de todos los elementos de la imagen las realiza el sistema, el usuario introduce las coordenadas del contorno del objeto a reconocer, por medio del mouse para lograr el reconocimiento del objeto.

Las diferencias respecto a lo que se desarrollo, es que el sistema fue desarrollado con bibliotecas y software propietario, además de que el proceso de reconocimiento sería totalmente realizado por el sistema.

”Sistema traductor para el reconocimiento del alfabeto dactilológico” [3]. Este trabajo consiste en interpretar y reconocer signos procedentes de imágenes del alfabeto dactilológico completo, es decir todas las letras. Utiliza 4 etapas denominadas procesamiento, segmentación, vectores de características y clase del objeto para aislar regiones de interés de cada postura de la mano. Además de desarrollar una metodología de preprocesamiento en donde las imágenes sean obtenidas a partir de una webcam. Este software es propietario y no es claro el tipo de métodos matemáticos que emplea

La diferencia con éste trabajo es que en mi proyecto no se toman en cuenta las letras J y Z debido a que éstas incluyen movimiento, y se asume que se cuenta con las imágenes almacenadas para realizar el procesamiento.

DITS: programa informático para el aprendizaje del código dactilológico'' [4]. Éste programa es una herramienta de ayuda técnica a personas discapacitadas, está compuesto por 3 módulos de aprendizaje, el primero inicia introduciendo una palabra para ser interpretada al lenguaje dactilológico, en el segundo módulo el usuario introduce una frase que deberá ser traducida a imágenes que describan la oración. En el tercer módulo muestra una actividad hablada y es interpretada por el lenguaje de señas.

Éste programa no está realizado con herramientas libres y no permite acceso al código fuente, además requiere como plataforma de trabajo Microsoft Windows.

2.2 Referencias internas

En la UAM-A se han realizados un proyecto que utiliza el tratamiento de imágenes al sistema propuesto:

"Clasificador de objetos en banda infinita por medio de procesamiento de imágenes" [5]. Este trabajo es interno de la UAM, en él se propone clasificar objetos con características determinadas en altura y profundidad, por medio del procesamiento digital de imágenes. Se hace uso de hilos para la sincronización y técnicas de concurrencias de procesos.

El proceso de desarrollo es acceder al hardware para seleccionar el dispositivo que tomará la imagen del fondo del objeto a identificar, después se realiza una resta de éstas para que el sistema separe al objeto del fondo y así proceder al reconocimiento de la imagen.

Éste es desarrollado en sistema operativo Microsoft Windows [6] en lenguaje de programación C#. El trabajo no está orientado al aprendizaje del alfabeto dactilológico.

2.3 Marco teórico

En esta sección se describen los temas de procesamiento de imágenes, transformada Wavelet discreta y redes neuronales que fueron usados en el desarrollo del presente trabajo.

2.3.1 Imagen

Una imagen corresponde a una función f (x, y) en donde (x, y) son las coordenadas de cada elemento de la imagen o pixel y el valor de la función representa la intensidad del pixel. Cuando la intensidad varía entre 0 y 255 su valor puede almacenarse en un byte así que un formato de archivo comúnmente usado bajo este esquema es PGM (Portable Pixel Map) que incluye un encabezado simple parecido al siguiente:

P5 Número mágico, PGM un byte por pixel
#ComentarioX La dimensión horizontal (ancho) Y La dimensión vertical (alto)
255 El valor maximo de los pixeles
pixel0 pixel1 pixel2 ... pixel(ancho-1) ...
.
.
.
pixel(ancho x alto-1)

2.3.2 Preprocesamiento de la imagen

Los campos del procesamiento digital de imágenes refieren al procesamiento digital por medio de una computadora. Teniendo en cuenta que una imagen digital se compone de una infinidad de números de elementos, cada uno de los cuales tiene una particular localización y evaluación. Estos elementos se denominan elementos de imagen, píxel, y pixeles. Pixel es el término más utilizado para denota los elementos de imagen digital.

No hay un acuerdo general entre los autores con respecto a dónde termina el procesamiento de la imagen y donde comienzan otras áreas relativas como el análisis en imágenes y la visión por computadora.

Algunas, veces se hace una distinción mediante la definición de procesamiento de imagen como una disciplina en el que tanto la entrada de una salida de un proceso son las imágenes, donde se extraen caracteres individuales(segmentación) que se describen por medio de una procesamiento adecuado de computadora(análisis de características) y posteriormente los caracteres se reconocen.

2.3.2.1 Transformada wavelet discreta

Los wavelets son funciones matemáticas asociadas a una función de escalamiento que tiene la propiedad de ser descrita como una versión escalada y desplazada de sí misma. Quizá el wavelet más sencillo de estudiar es el wavelet de Haar cuya expresión matemática es:

$$\psi(t) = \begin{cases} 1 & 0 \le t < 1/2, \\ -1 & 1/2 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Y que está asociado con la función de escalamiento de Haar cuya expresión matemática es:

$$\phi(t) = \begin{cases} 1 & 0 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

La transformada wavelet de Haar, o simplemente transformada Haar, realiza una descomposición de un vector o señal en una parte burda y un detalle. La parte burda de una señal discreta como es una imagen es obtenida a través de la aplicación de un filtro pasabajas que básicamente realiza el promedio de cada dos muestras del vector dado. El detalle es obtenida con un filtro pasaaltas que básicamente obtiene la diferencia entre cada par de muestras[1]. Sean s0 y s1 dos muestras de un vector a transformar.

[1] Se pude verificar que al introducir un factor de escalamiento de $\frac{2}{\sqrt{2}}$ se conserva la energía en el proceso de transformación

La componente de baja frecuencia está dada por $a = \frac{s0+s1}{2}$ en tanto que la componente de alta frecuencia está determinada por $c = \frac{s0-s1}{2}$ La transformada inversa Haar permite una reconstrucción perfecta que se obtiene al combinar los coeficientes de transformación de la siguiente manera $a + c = \frac{(s+s1)+(s0-s1)}{2} = s_0 \;\; y \;\; a - c = \frac{(s+s1)+(s0-s1)}{2} = s_1$ Las operaciones de filtrado pueden verse como una multiplicación matricial del [$s_0 s_1$] con

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Para vectores con n muestras se usa la matriz de transformación de n x n, por ejemplo, para el vector de entrada [$s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7$] la transformación de Haar genera:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \bullet \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Figura 2.3.2.1 Transformación de Haar

En donde los coeficientes a_i son coeficientes de baja frecuencia y los coeficientes c_i son de alta frecuencia. Los coeficientes se acomodan colocando primero los de baja frecuencia y luego los de alta frecuencia.

La transformación se sigue aplicando a las primeras $\frac{n}{2}$ muestras de baja frecuencia lo que da lugar a $\frac{n}{2}$ muestras de baja y alta frecuencia en el segundo nivel de transformación. La transformada wavelet en 2D consiste en aplicar primero la transformación en dirección horizontal (o vertical) sobre los vectores de una matriz bidimensional seguida de una transformación vertical (horizontal) sobre los vectores de la matriz previamente transformada en una dimensión.

2.3.2.2 Red neuronal

Una neurona [8] es una unidad de procesamiento que se activan o desactivan como respuesta a estímulos que llegan a través de los sentidos humanos. Las tres partes fundamentales de una neurona son su entrada, su salida y la capacidad de procesamiento. El cerebro humano tiene millones de ellas conectadas mediante sinapsis que, de acuerdo con el postulado de Hebb, se fortalecen cuando dos neuronas se activan simultáneamente y se debilitan cuando una neurona se activa y la otra inhibe su activación. Por consiguiente se hace referencia a redes de neuronas o redes neuronales.

Una neurona artificial es un modelo matemático inspirado en las neuronas biológicas. Uno de los modelos más conocidos, pero desde luego no el único, es el que propuso Rosenblat cuyas partes son:

- Entradas. Un vector que representa los rasgos característicos de objetos que procesa la neurona. Incluye un elemento BIAS con valor unitario. Se denotan como X = x0, x1, x2, ...
- Pesos sinápticos. Un vector con números reales que representan las sinapsis del modelo biológico. Se denotan como W = w0, w1, w2 , ...
- Sumador. Realiza una suma aritmética.
- Potencial de activación. Es la suma ponderada resultado del producto punto del vector de entradas con el vectores de pesos sinápticos, se denota como v =<X W>
- Función de activación. Una función matemática que indica si la neurona se activa o no de acuerdo al valor del potencial de activación.
- Salida. Un valor que indica si la neurona artificial se activa ante los valores del vector de entrada o no. Típicamente toma valores de 0 y 1 para indicar la decisión. Se denota por la variable o.

El ejemplo siguiente muestra a un perceptrón modificado que usa una función sigmoidal en lugar de una función escalón como la propuesta originalmente por Rosenblat. Se muestran los valores de sus pesos sinápticos que almacenan (memorizan) la función lógica OR. Figura 3.2.2.2.

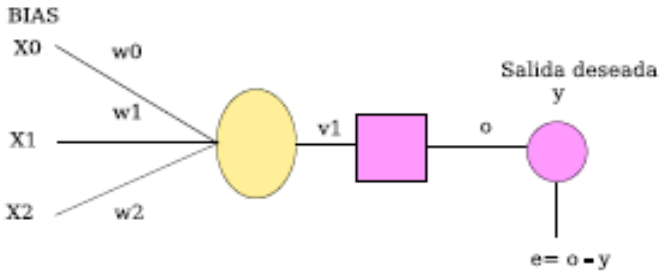

Figura 2.3.2.2.Perceptrón con función sigmoidal

Cuando se presenta a la entrada el vector [1 0 0] el potencial de activación es v = w_0 y la salida de la función de activación es $o = \frac{1}{1+\exp(-w_0)}$. Si w_0 =-14 o= 0,0000008 que nos da una buena aproximación a cero. Los demás casos del vector de entrada Incluyendo pequeñas variaciones en los valores se muestran en la tabla.

En la tabla 2.3.2.2 se observa cómo las variaciones en los vectores de entrada no generan cambios abruptos en la salida de perceptrón, lo cual manifiesta la propiedad de tolerancia a ligeras variaciones (ruido), algo que es difícil, sino es que imposible, de lograr con circuitos digitales.

0	0	0.0000008
0	1	0.9999991
1	0	0.9999991
1	1	0.9999999
0.1	0.2	0.0036842
0.8	0.9	0.9999999

Tabla 2.3.2.2 tabla de variaciones

Una red neuronal puede usarse como memoria asociativa, que a diferencia de las memorias indexadas no reciben una dirección para obtener un dato almacenado en cierta localidad, sino que aceptan como entrada las características de un objeto y ofrecen a la salida el objeto que más se parezca a otro que se le ha presentado previamente.

En el caso de la tabla se provee a la entrada vectores similares a [1 0 0] y la salida es cercana a cero, mientras que vectores parecidos a [1 1 1] producen una salida cercana a uno. Esto significa que el perceptrón actúa como un sistema clasificador de los vectores de entrada, la clase C_1 que agrupa a los vectores parecidos a [1 0 0] y la clase C_2 que agrupa a los vectores parecidos a [1 1 1].

Cuando se conectan las salidas de varios perceptrones como entradas de otros se obtiene una red de perceptrones. Según el Teorema de Aproximación Universal un conjunto de perceptrones es capaz de aproximar funciones matemáticas complicadas sin necesidad de expresarlas explícitamente, sino que la salida está dada por la evaluación en cascada de la salida de varios perceptrones que entran a otros perceptrones, cuyas salidas vuelven a ser entradas de otra capa de perceptrones.

Considerando el teorema de aproximación universal se parte de la premisa de que si existe una función matemática que permita clasificar imágenes de las letras del alfabeto dactilológico, una red de perceptrónes será capaz de aproximarla.

Se distinguen varios procesos al trabajar con redes neuronales.

- Escalamiento. Consiste en llevar los valores de los datos de entrada al intervalo [0, 1]

- Entrenamiento. Consiste en poner a la entrada de la red los datos por aprender y que dan lugar a un ajuste de los pesos sinápticos.

- Aprendizaje. Consiste en ajustar los valores de los pesos sinápticos para que se minimice el error entre la salida deseada y la obtenida.

- Prueba. Consiste en probar el desempeño de la red neuronal con vectores de entrada que no causan ajuste de los pesos sinápticos y que sin embargo debe aprenderse su salida correcta.

- Generalización. Es la capacidad que adquiere una red neuronal para proveer salidas aceptables ante vectores de entrada que no se usaron en el entrenamiento.

El Teorema del convergencia del perceptrón indica que al presentarle los vectores (muestras) de entrada en forma repetitiva (entrenamiento) y ajustando los pesos en forma proporcional al error de aproximación conlleva a que los pesos sinápticos adquieran valores que minimicen el error global sobre todos los vectores de entrada.

Para entrenar una red de perceptrónes en multicapa (Multilayer perceptrons-MLP) existen varios métodos, entre los que se encuentra el algoritmo de Retropagación (Backpropagation) y es el que se ha usado en este trabajo.

CAPÍTULO 2. ANTECEDENTES

Capítulo 3
Descripción y especificación Técnica

3.1 Descripción del sistema de reconocimiento.

El proyecto originalmente se componía de cuatro módulos: segmentación, extracción de características, reconocimiento y presentación de resultados, según se ilustra en la Figura 3.1.

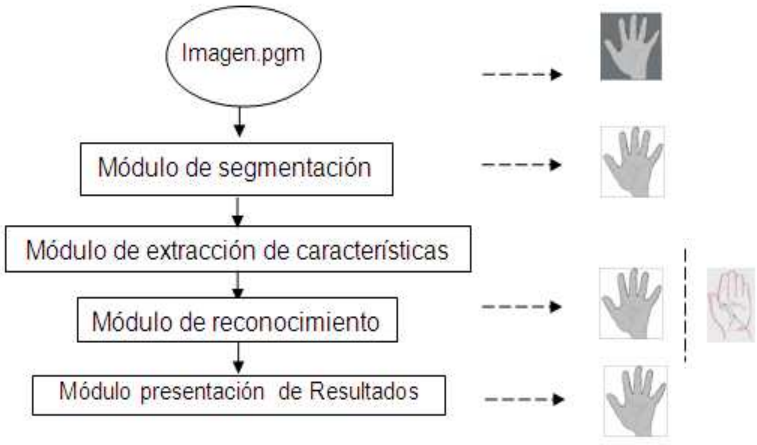

Figura 3.1: Módulos del proyecto

Durante la realización del proyecto se consideró innecesaria implementar el módulo de segmentación.

A grandes rasgos el proceso de reconocimiento inicia cuando el usuario selecciona una imagen, almacenada previamente en un conjunto de imágenes de 128 x 128 pixeles. Dicha imagen deberá estar en una escala de grises y en formato PGM con el fin de facilitar el tratamiento. La imagen será procesada con la finalidad de obtener datos que permitan lograr el reconocimiento de alguna de las letras del alfabeto. Posteriormente los resultados serán presentados en una interfaz gráfica en donde se mostrará tanto la imagen fuente como la imagen más parecida.

A continuación se describen cada uno de los módulos del sistema de reconocimiento de imágenes del alfabeto dactilológico.

3.2 Módulos principales

3.2.1 Segmentación de la imagen.

Inicialmente se consideró que sería conveniente segmentar imágenes (identificar regiones) para facilitar el proceso de reconocimiento, sin embargo a lo largo del desarrollo del proyecto se decidió prescindir de él porque el uso de wavelets permitió trabajar con toda la imagen y reducir el tamaño de las mismas, además de que se obtenían resultados favorables.

3.2.2 Extracción de características.

Las imágenes del alfabeto dactilológico se pueden exportar (de JPEG, PNG, GIF, etc) al formato PGM a fin de facilitar su procesamiento. El procesamiento incluye aplicar la transformada wavelet discreta que recibe como entrada los valores de pixel en n vectores de dimensión n (matriz de n x n). Una de las propiedades de la transformada wavelet es que no solo realiza un mapeo al dominio de la frecuencia como la transformada de Fourier, sino que hace un mapeo a dominio tiempo-frecuencia con lo cual al aplicar repetidamente la transformación de una imagen se genera un efecto piramidal en donde los niveles más altos concentran la energía y mantienen una versión a la

imagen pero con dimensión de nivel de transformación lo cual permite usar una versión más pequeña de la imagen y conservar una gran cantidad de información, en otras palabras, los coeficientes de transformación de los niveles más altos reúnen rasgos característicos de la imagen original, al mismo tiempo que el posible ruido es filtrado en las componentes de alta frecuencia. Por lo que al aplicar Wavelet a las imágenes de 128 X 128 obtuvimos otras.

Otro paso que se realiza sobre las imágenes es rotarlas usando el operador de transformación. Cada pixel ubicado en la coordenada (x, y) es llevado a su nueva posición (x^i, y^i) en donde $x^i = \cos\Theta - \text{sen}\Theta$ y $y^i = \cos\Theta + \text{sen}\Theta$. El ángulo Θ está en radianes que se puede convertir a grados con la ecuación grados = $\frac{360\Theta}{2\pi}$.

La idea de rotar cada imagen es tener variaciones de una misma letra y darle información a una red neuronal artificial a fin de que una vez entrenada sea capaz de manejar invariancia a la rotación.

La metodología parte de obtener versiones de cada imagen, se rota -50,-40,-20,-30,-10,0,10,20,30,40,50 grados, así que porcada imagen se generan 11 vectores etiquetados bajo el mismo tipo, lo cual permite hacer invariancia a la rotación en el reconocimiento.

3.2.3 Reconocimiento de imágenes.

En el caso del reconocimiento de las letras del alfabeto dactilológico se requiere que la red neuronal clasifique los vectores con los coeficientes de transformación wavelet en 24 clases, la primera clase CA son 11 vectores que representan variantes rotadas de la letra A, una segunda clase CB son once vectores que representan variantes rotadas de la letra B, y así sucesivamente hasta la letra Y. Las letras J y Z no son consideradas en este trabajo porque no se pueden almacenar como imagen ya que involucran movimientos de los dedos.

En este módulo se emplea una red neuronal, esto permite hacer búsqueda por similitud más que por direccionamiento y con ello dada una imagen en la imagen en el sistema de reconocimiento es posible obtener como salida la letra correcta que representa.

Posteriormente se obtuvieron los pesos arrojados por la red neuronal, y estos se aplicaron en un algoritmo. Al aplicar estos pesos se consiguió algo muy interesante, es decir se independizo de alguna forma la implementación de un algoritmo de red neuronal en la aplicación. Logrando con una serie de operaciones y lecturas de arreglos obtener la imagen más coincidente con la introducida en la aplicación.

Para obtener la imagen mas coincidente se realizo una segmentación de 24 valores entre [0,1] con la finalidad de ajustar la salida del algoritmo a la literal correspondiente.

3.2.4 Presentación de resultados.

Finalmente se tiene el módulo de presentación de resultados, el cual consiste en una interfaz gráfica donde se muestra la imagen que el usuario seleccionó y la más parecida a la imagen original. Con ello, el usuario podrá apreciar las imágenes, compararlas y verificar que el reconocimiento es correcto, así como la flexibilidad en la rotación de la imagen de acuerdo a una postura de la mano. Las imágenes capturadas se almacenan en un directorio para que sean leídas por el software de reconocimiento.

El diseño de la interfaz tiene como objetivo ser sencilla por lo que sólo tiene dos botones uno para cargar la imagen usando la ventana de selección (ver figura 6.2) en la aplicación y otro para salir del sistema, según se ilustra en la figura 3.2.4.

Figura 3.2.4: Interfaz de la aplicación

Al cargar la imagen se comienza por extraer las características de la imagen, es decir se obtienen los coeficientes wavelet y se inicia con el proceso de reconocimiento.

3.3 Especificación técnica.

El uso de la aplicación es sencillo, por lo que se deben prever algunos aspectos antes de su utilización para obtener resultados fiables.

- Las imágenes deberán tener un tamaño de 128 x 128 pixeles.
- Las imágenes deben tener una extensión pgm.
- El nombre de la imagen deber comenzar con la literal a la que haga referencia por ejemplo:

a.pgm, Aalgo.pgm, anombre.pgm

- Las imágenes deberán estar almacenadas en el directorio de la aplicación.
- Si se desea ocupar solo el teclado deber

Presionar:

Alt+c para cargar la imagen

Alt+s para salir de la aplicación

CAPÍTULO 3. DESCRIPCIÓN Y ESPECIFICACIÓN TÉCNICA

Capítulo 4
Experimentos

Se consideraron 24 de las 26 letras del abecedario dactilológico mostradas a continuación:

Figura 4.1: Alfabeto Dactilológico

Se obtuvieron 24 imágenes con dimensión 128 x 128 pixeles en formato JPG que fueron exportadas a formato PGM. Luego se aplicó la transformada wavelet con 3 niveles de transformación, esto generó una pirámide cuya componente de más baja frecuencia (coeficientes que concentran la energía) tiene 16 x 16 coeficientes con valores reales. Estos valores de los coeficientes fueron llevados al intervalo [0, 1]. A fin de tener una representación visual de esos coeficientes se multiplicaron por un factor de 255 y se almacenaron como si fueran imágenes PGM y se muestran a continuación.

Se tienen 24 x 11 = 264 vectores para las letras y sus variantes rotadas, de los cuales se tomaron 211 de ellos en forma aleatoria para realizar el entrenamiento. A la presentación de las 211 muestras de entrenamiento se le llama época. Cada 100 épocas se presentan las 57 muestras restantes como muestras de prueba y con ellas no se ajustan los pesos sinápticos. El entrenamiento se detiene cuando el error promedio de prueba es mínimo porque indica la que la red ha alcanzado la máxima capacidad de generalización. A este proceso se le conoce como paro temprano con validación cruzada.

Para realizar el entrenamiento se usó el software Data Engine. El software requiere básicamente de 3 archivos: Entrenamiento.dat, Prueba.dat, Recall.dat

El archivo Entrenamiento.dat almacena 258 renglones en donde cada renglón tiene 257 columnas. Las primeras 256 columnas corresponden a los coeficientes de de transformación wavelet y la última columna tiene el valor $\frac{ASCII\,(letra)-65}{24}$ en donde ASCII (letra) es el valor en decimal de las diferentes letras a reconocer (etiquetas numéricas de las clases CA, CB,. . ., CY).
El archivo Prueba.dat almacena 28 renglones en donde cada renglón tiene 257 columnas. Las primeras 256 corresponden a los coeficientes de transformación wavelet y la última columna tiene el valor $\frac{ASCII\,(letra)-65}{24}$ en donde ASCII (letra) es el valor en decimal de las diferentes letras a reconocer (etiquetas numéricas de las clases CA , CB, … , CY).

El archivo Recall.dat almacena 264 renglones en donde cada renglón tiene los 256 coeficientes de transformación wavelet. No se requiere la última columna de etiqueta-miento porque este archivo es usado para presentar los vectores de entrada y obtener las salidas usando al red neuronal entrenada.

El software Data Engine [9] provee un archivo RecallOut.dat en donde usa el archivo Recall.dat y le coloca una última columna con los valores aproximados. Esto permite hacer comparaciones con los valores teórico es esperados. Para ello se crea un archivo Todos.dat en donde se incluyen los 264 vectores de entrenamiento con 257 columnas, en donde las primeras 256 son coeficientes de transformación y la última es la etiqueta de la clase de letra en forma numérica (CA = 0,0; CB = 0,038; CC = 0,076, etc.)

4.1 Aplicación software.

El programa fue escrito en QT/C/C++, debido a su portabilidad y universalidad en el área del cómputo.

El desarrollo de la interfaz fue elaborado en el lenguaje de programación QT, utilizando el Qtcreator en su versio 4.6.2. La unión de los algoritmos y los módulos esta “alambrada” mediante clases desde la interfaz para facilitar su integración.

4.2 Hardware y software usado.

La aplicación se implementó, en una Laptop Acer Aspire 4320, con las siguientes características:

- Procesador: Intel celeron (1.73 Ghz)
- Memoria RAM: 512 SDRAM DDR2 de doble canal a 533 MHz
- La aplicación fue desarrollada en:

GNU/Linux Distribución Ubuntu 8.04 Se utilizo QT y C/C++ para desarrollar la interfaz y los algoritmos.

CAPÍTULO 4. EXPERIMENTOS

Capítulo 5
Tecnologías Utilizadas

5.1 Lenguajes de programación.

En esta sección se describen brevemente los lenguajes de programación y tecnologías que se han utilizado en el desarrollo del proyecto.

5.1.1 C++.

El lenguaje C++ es un lenguaje imperativo orientado a objetos derivado del C diseñado a mediados de los años 1980 por Bjarne Stroustrup; desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Se ha señalado, C++ no es un lenguaje orientado a objetos puro.

C++ introduce nuevas palabras clave y operadores para manejo de clases, algunas de sus extensiones tienen aplicación fuera del contexto de programación con objetos (fuera del ámbito de las clases), de hecho, muchos aspectos de C++ que pueden ser usados independientemente de las clases.

5.1.2 Qt.

El desarrollo de la Intefaz fue en QT, utilizando Qtcreator. Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Es producido por la división de software Qt de Nokia [10], que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.

Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación a través de bindings. Qt cuenta actualmente con un sistema de triple licencia: GPL v2/v3 para el desarrollo de software de código abierto y software libre, la licencia de pago QPL para el desarrollo de aplicaciones comerciales, y a partir de la versión 4.5 una licencia gratuita pensada para aplicaciones comerciales, GPL.

Qt se encuentra disponible para sistemas tipo unix con el servidor gráfico X Window System (Linux, BSDs, Unix), para Apple Mac OS X, para sistemas Microsoft Windows.

5.1.3 Ventajas de C++ y Qt como lenguaje de programación

Adicionalmente se pueden destacar las siguientes características del C++:

- Lenguaje de programación orientado a objetos.
- Lenguaje muy didáctico, gracias a este lenguaje puedes aprender muchos otros lenguajes con gran facilidad.
- Alta productividad

- Soporte de caracteres internacionales
- Es muy potente en lo que se refiere a creación de sistemas complejos, un lenguaje muy robusto.
- Rápido ensamblaje de aplicaciones
- Menor costo y rápida implementación

Características de QT:

- Qt es completamente gratuito para aplicaciones de código abierto.
- Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones.
- Se pueden desarrollar ricas aplicaciones gráficas.
- Soporte de caracteres internacionales.
- Incluye soporte de nuevas tecnologías como OpenGL, XML, Bases de Datos, programación para redes.

CAPÍTULO 5. TECNOLOGÍAS UTILIZADAS

Capítulo 6
Resultados de la aplicación

Luego de una larga investigación y desarrollo, se logró terminar la aplicación; tomando en cuenta el trabajo que costo desarrollar la aplicación, es para mi un gran logró personal.

A continuación se muestra el resultado que se obtuvo al desarrollar y probar la aplicación.

6.1 Resultados de la aplicación

Se logró minimizar el error de entrenamiento a un valor de 0,0001 y el error de prueba a un valor de 0,0015. Para realizar la comprobación del aprendizaje se colocaron los 264 vectores a la entrada de la red neuronal y para cada uno de ellos se obtiene la salida aproximada de la red. La salida obtenida es un valor real entre 0 y 1, en donde la letra A está identificada por el valor 0, la letra B por el valor $\frac{1}{24}$, la letra C por el valor $\frac{2}{24}$ y así sucesivamente hasta la Y representada por el valor $\frac{24}{24}$ A este proceso se le llama RECALL y el error promedio obtenido fue 0.0001. En la figura 6.1 se muestra la grafica de entrenamiento que se obtuvo.

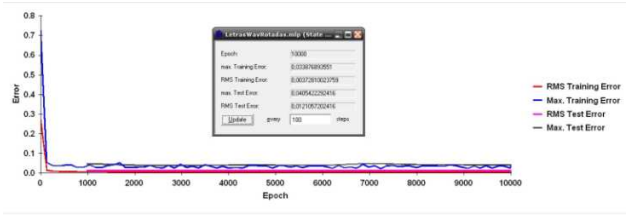

Figura 6.1: Gráfica de entrenamiento

CAPÍTULO 6. RESULTADOS

Como en toda clasificación se tiene un error promedio por ello cabe mencionar que se equivoca al clasificar algunas imágenes, mostrando mayor discrepancia al clasificar

la letra Y y X, esto se debe a que ambas imágenes tienen coeficientes característicos similares. La interpretación de estas letras no es parecida, por lo que se abre la brecha para el uso de otras técnicas que mejoren los datos característicos de las imágenes en un trabajo futuro.

La forma de trabajo de la aplicación es sencilla; al cargar una imagen se aplica transformada wavelet de Haar a la imagen, obteniendo de esta forma los datos que se usarán para identificar la imagen coincidente con la ingresada.

A continuación se muestran algunos ejemplos para ilustrar el funcionamiento de la aplicación.

Figura6.1 inicio de la aplicación

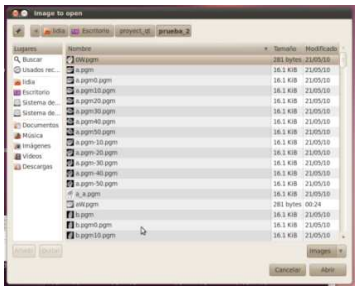

Figura6.2 cargado de la imagen

Figura6.3: Imagen de la literal A reconocida.

Figura6.4: Imagen de la literal L reconocida

6.2 Trabajo a futuro.

Este trabajo da pie a explorar el uso de otros wavelets como los Daubechies y comparar su desempeño con el wavelet de Haar usado en este trabajo.

Así como probar la metodología descrita en este trabajo para realizar reconocimiento de formas geométricas (triángulos, cuadrados, elipses, estrellas, etc.). Probar la metodología descrita en este trabajo para realizar reconocimiento de imágenes con huellas digitales.

La aplicación desarrollada puede llevarse a un mayor nivel capturando imágenes en tiempo real usando una cámara digital y realizar el reconocimiento de estas, para que acto seguido pueda transmitirse en forma de texto (y o como imagen) por Internet y desplegar en otra computadora la imagen de la letra del alfabeto dactilológico correspondiente, lo cual permite hacer compresión de imagen de una dimensión de 128 x 128 pixeles a un único byte.

CAPÍTULO 6. RESULTADOS

Capítulo 7
Manual de usuario

Para poder usar la aplicación se deberá instalar el motor base de Qt. El motor de base es un paquete llamado Qt, generalmente se obtiene al descargar una versión de Qt. El desarrollo se llevó acabó utilizando QtCreator versión 4.6.2 que se puede descargar de la pagina oficial.

Esta aplicación fue desarrollada con la versión GNU/Ubuntu por lo que el modo de instalación en otros sistemas operativos podría variar.

7.1 Modo de Ejecución

Para usar la aplicación se recomienda compilar el programa, ya sea desde una interfaz de Qt presionando CTRL + R para el caso de QtCreator versión 4.6.2. La forma de ejecución depende de la versión instalada.

Desde línea de comandos se puede compilar el código introduciendo en una terminal dentro de la carpeta de la aplicación:

- qmake -project
- qmake
- make

Para ejecutar escribir

./prueba_2

Una vez mostrada la interfaz se da clic en el botón cargar, para ingresar la imagen, posteriormente la aplicación mostrará la imagen más coincidente con está.

En caso de no encontrar ninguna coincidencia, se mostrará un mensaje con la leyenda imagen no reconocida.

Es importante no manejar un tamaño mayor a 128 x 128 pixeles, debido a que la aplicación se diseño para trabajar con tamaños no mayores a estos valores.

Ejemplo de Ejecución

Este ejemplo se ilustra en QtCreator versión 4.6.2.

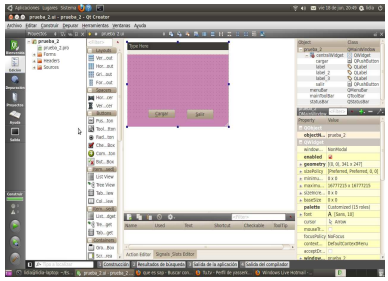

Figura7.1 Aplicación en QT.

Para iniciar el proceso se presiona el botón de ejecutar o bien CTRL + R. De inmediato aparecerá la pantalla de la aplicación. Dar clic en el botón cargar.

Figura7.2 Pantalla de la aplicación

Seleccionar la imagen a reconocer

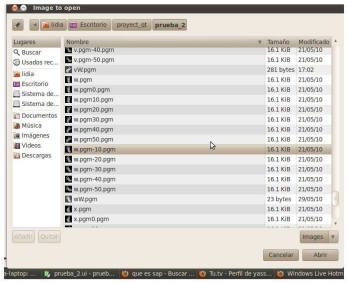

Figura7.3 Cargar la imagen.

Posteriormente se muestra el resultado.

Figura7.4 Presentación de resultados.

Apéndice

Código fuente

Documentación de las clases

Referencia de la Clase Pesos

#include <Pesos.h>

Métodos públicos

- Pesos ()
- ~Pesos ()
- char leer ()
- void leer2 ()
- char sumatoria ()
- float sigmoidal (float)
- char desicion (float)

Atributos privados

- char buff [4096]
- char buff_1 [10096]
- int fd_rd
- int rd
- int fd_rd_2
- int rd_2
- float Wa [257][8]
- float Wb [9]
- float Wc [8]
- float Wd [9]
- float X [259]
- double f [4096]
- float ff

Documentación del constructor y destructor

- Pesos::Pesos ()
- Pesos::~Pesos ()

Documentación de las funciones miembro

- char Pesos::leer ()

Gráfico de llamadas para esta función:

- void Pesos::leer2 ()
- char Pesos::sumatoria ()

Gráfico de llamadas para esta función:

- float Pesos::sigmoidal (float v)
- char Pesos::desicion (float suma)

Documentación de los datos miembro

- char Pesos::buff[4096] [private]
- char Pesos::buff_1[10096] [private]
- int Pesos::fd_rd [private]
- int Pesos::rd [private]
- int Pesos::fd_rd_2 [private]
- int Pesos::rd_2 [private]

- float Pesos::Wa[257][8] [private]
- float Pesos::Wb[9] [private]
- float Pesos::Wc[8] [private]
- float Pesos::Wd[9] [private]
- float Pesos::X[259] [private]
- double Pesos::f[4096] [private]
- float Pesos::ff [private]

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Codigo_fuente_apendice/Pesos.h
- Codigo_fuente_apendice/Pesos.cpp

Documentación de los datos miembro

- char Pesos::buff[4096] [private]
- char Pesos::buff_1[10096] [private]
- int Pesos::fd_rd [private]
- int Pesos::rd [private]
- int Pesos::fd_rd_2 [private]
- int Pesos::rd_2 [private]
- float Pesos::Wa[257][8] [private]

- float Pesos::Wb[9] [private]
- float Pesos::Wc[8] [private]
- float Pesos::Wd[9] [private]
- float Pesos::X[259] [private]
- double Pesos::f[4096] [private]
- float Pesos::ff [private]

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Codigo_fuente_apendice/Pesos.h
- Codigo_fuente_apendice/Pesos.cpp

Referencia de la Clase Ui::prueba_2

#include <ui_prueba_2.h>
Diagrama de herencias de Ui::prueba_2
Diagrama de colaboración para Ui::prueba_2:

La documentación para esta clase fue generada a partir del siguiente fichero:
Codigo_fuente_apendice/ui_prueba_2.h

Referencia de la Clase prueba_2
#include <prueba_2.h>
Diagrama de colaboración para prueba_2:

Slots públicos

- void algo ()
- void algo_2 ()
- void algo_3 ()
- void algo_4 ()
- void algo_5 (char)
- void algo_6 (QString)

Métodos públicos

- prueba_2 (QWidget *parent=0)
- ~prueba_2 ()

Atributos públicos

- PgmRotate * r
- PgmRotate * r2
- Haar2D * h
- Pesos * P

Métodos protegidos

- void changeEvent (QEvent *e)

Atributos privados

- Ui::prueba_2 * ui

Documentación del constructor y destructor

prueba_2::prueba_2 (QWidget * *parent* = 0)

Gráfico de llamadas para esta función:

prueba_2::~prueba_2 ()

Documentación de los datos miembro

PgmRotate* prueba_2::rPgmRotate* prueba_2::r2Haar2D* prueba_2::hPesos* prueba_2::PUi::prueba_2* prueba_2::ui [private]

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- Codigo_fuente_apendice/prueba_2.h
- Codigo_fuente_apendice/prueba_2.cpp

Referencia de la Clase Ui_prueba_2

#include <ui_prueba_2.h>

Diagrama de herencias de Ui_prueba_2

Métodos públicos

- void setupUi (QMainWindow *prueba_2)
- void retranslateUi (QMainWindow *prueba_2)

Atributos públicos

- QWidget * centralWidget
- QPushButton * salir
- QLabel * label
- QPushButton * cargar
- QLabel * label_2
- QLabel * label_3
- QMenuBar * menuBar
- QToolBar * mainToolBar
- QStatusBar * statusBar

Documentación de las funciones miembro

void Ui_prueba_2::setupUi (QMainWindow * *prueba_2*) [inline]

Gráfico de llamadas para esta función:

void Ui_prueba_2::retranslateUi (QMainWindow * *prueba_2*) [inline]

Documentación de los datos miembro

QWidget* Ui_prueba_2::centralWidgetQPushButton* Ui_prueba_2::salirQLabel* Ui_prueba_2::labelQPushButton* Ui_prueba_2::cargarQLabel* Ui_prueba_2::label_2QLabel* Ui_prueba_2::label_3QMenuBar* Ui_prueba_2::menuBarQToolBar* Ui_prueba_2::mainToolBarQStatusBar* Ui_prueba_2::statusBar

La documentación para esta clase fue generada a partir del siguiente fichero:

Codigo_fuente_apendice/ui_prueba_2.h

Documentación de archivos

Referencia del Archivo Codigo_fuente_apendice/main.cpp

```
#include <QtGui/QApplication>
#include "prueba_2.h"
```

Dependencia gráfica adjunta para main.cpp:

Funciones

- int main (int argc, char *argv[])

Documentación de las funciones

int main (int *argc*, char * *argv*[])

Referencia del Archivo Codigo_fuente_apendice/moc_prueba_2.cpp

```
#include "prueba_2.h"
```

Dependencia gráfica adjunta para moc_prueba_2.cpp:

Variables

- static QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_prueba_2 []
- static const char qt_meta_stringdata_prueba_2 []

Documentación de las variables

QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_prueba_2[] [static]

Valor inicial: {

```
    4,
    0,
    0,    0,
    6,   14,
    0,    0,
    0,    0,
    0,    0,
    0,
    0,

   10,    9,    9,    9, 0x0a,
   17,    9,    9,    9, 0x0a,
   26,    9,    9,    9, 0x0a,
   35,    9,    9,    9, 0x0a,
   44,    9,    9,    9, 0x0a,
   57,    9,    9,    9, 0x0a,

    0
}
```

const char qt_meta_stringdata_prueba_2[] [static]

Valor inicial: {

```
  "prueba_2\0\0algo()\0algo_2()\0algo_3()\0"
  "algo_4()\0algo_5(char)\0algo_6(QString)\0"
}
```

Referencia del Archivo Codigo_fuente_apendice/Pesos.cpp

```
#include "Pesos.h"
```

Dependencia gráfica adjunta para Pesos.cpp:

Documentación de las definiciones

#define __Pesos__ 1

#define numMax 4096

Referencia del Archivo Codigo_fuente_apendice/prueba_2.cpp

```
#include "prueba_2.h"
#include "ui_prueba_2.h"
#include <QPixmap>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <qwidget.h>
#include <QString>
#include <QImage>
#include <QDir>
```

Dependencia gráfica adjunta para prueba_2.cpp:

Referencia del Archivo Codigo_fuente_apendice/Pesos.h

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <fcntl.h>
#include <string>
#include <iostream>
#include <fstream>
```

Dependencia gráfica adjunta para Pesos.h:

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

Clases

- class Pesos

Definiciones

- #define __Pesos__ 1
- #define numMax 4096

Documentación de las definiciones

#define __Pesos__ 1

#define numMax 4096

Referencia del Archivo Codigo_fuente_apendice/prueba_2.cpp

```
#include "prueba_2.h"
#include "ui_prueba_2.h"
#include <QPixmap>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <qwidget.h>
#include <QString>
#include <QImage>
#include <QDir>
```

Dependencia gráfica adjunta para prueba_2.cpp:

Referencia del Archivo Codigo_fuente_apendice/prueba_2.h

```
#include <QMainWindow>
#include "PgmRotate.h"
#include "Haar2D.h"
#include "Pesos.h"
```

Dependencia gráfica adjunta para prueba_2.h:

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

Namespaces

- namespace Ui

Clases

- class prueba_2

Pesos.cpp

```
#include "Pesos.h"
Pesos::Pesos(){}
Pesos::~Pesos(){}

char Pesos::leer(){
	int k=0,l=0;
	FILE *ptr;
	ptr=fopen("weigh_2.txt","r");
	fseek(ptr,0,0);
	for(l=0;l<8;l++){
		fscanf(ptr,"%f",&Wa[0][l]);
	}
	fscanf(ptr,"%f",&Wb[0]);
	Wd[0]=Wb[0];
	for(k=1;k<258;k++){
		for(l=0;l<8;l++){
			fscanf(ptr,"%f",&Wa[k][l]);

			if(k==257){
				Wc[l]=Wa[257][l];
			}
		}
	}
	for(k=1;k<9;k++){
		fscanf(ptr,"%f",&Wb[k]);
		Wd[k]=Wb[k];
	}
	fclose(ptr);
	leer2();
	char d=sumatoria();
	return d;
}
```

```
char Pesos::sumatoria(){
int k=0,l=0;
	float	sum1=0.0,sum2=0.0,sum3=0.0,sum4=0.0,sum5=0.0,sum6=0.0,sum7=0.0,sum8=0.0;
	float y1,y2,y3,y4,y5,y6,y7,y8;
	float v0,v1,v2,v3,v4,v5,v6,v7,v8,suma;
	char d=' ';
	int i=0;
	for(k=0;k<8;k++){
		Wa[257][k]=Wc[k];
		}
	for(i=0;i<258;i++){//calcula sumatorias en total 8 y multiplica por las X[i] entradas
		sum1+=Wa[i][0]*X[i];
		sum2+=Wa[i][1]*X[i];
		sum3+=Wa[i][2]*X[i];
		sum4+=Wa[i][3]*X[i];
		sum5+=Wa[i][4]*X[i];
		sum6+=Wa[i][5]*X[i];
		sum7+=Wa[i][6]*X[i];
		sum8+=Wa[i][7]*X[i];
	}
	y1=sigmoidal(sum1);//obtiene sigmoidal de cada sumatoria, en total 8
	y2=sigmoidal(sum2);
	y3=sigmoidal(sum3);
	y4=sigmoidal(sum4);
	y5=sigmoidal(sum5);
	y6=sigmoidal(sum6);
	y7=sigmoidal(sum7);
	y8=sigmoidal(sum8);
	printf("\n");
	v0=Wd[0]*1; //y0 multiplica bias*1
	v1=Wd[1]*y1;//multiplica las salidas de cada neurona(8)--> WB* las Y(sigmoidales_fi de y)
	v2=Wd[2]*y2;
	v3=Wd[3]*y3;
	v4=Wd[4]*y4;
	v5=Wd[5]*y5;
	v6=Wd[6]*y6;
	v7=Wd[7]*y7;
	v8=Wd[8]*y8;
	suma=v0+v1+v2+v3+v4+v5+v6+v7+v8;//suma los valores de las fi
	suma=suma;
	d=desicion(suma);
	return d;
}
void Pesos::leer2(){
	int k=0;
```

```
		char *t1;
		X[0]=1;
		fd_rd=open ("datos.txt",O_RDONLY);
		rd=read(fd_rd,buff,numMax);

		ff = atof (buff);
		X[1]=ff;
		k=strlen(buff);
	k=k/6;
		t1=strtok(buff,"\t");
		for(int i=0;i<k-2;i++){
			t1 = strtok(NULL, "\t");
			ff = atof (t1);
			X[i+2]=ff;

		}
		int p=0;
}
float Pesos::sigmoidal(float v){
		float w,x,y;
		x=-1*v;
		w=exp(x);
		y=1/(1+w);
		return y;
}
char Pesos::desicion(float suma){
		float
Vf[24]={0.060000,0.10000,0.140000,0.180000,0.220000,0.270000,0.310000,0.340000,0.380000,0.420000
,0.460000,0.520000,0.560000,0.600000,0.640000,0.680000
,0.720000,0.770000,0.810000,0.850000,0.889990,0.929990,0.959999,1};
float
Vi[24]={0,0.060001,0.010001,0.140001,0.180001,0.220001,0.270001,0.310001,0.340001,0.380001,0.420
001,0.460001,0.520001,0.560001,0.600001,0.640000,0.680001
,0.720000,0.770001,0.810001,0.850001,0.889991,0.929991,0.960000};
		char letra[24]={'a','b','c','d','e','f','g','h','i','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y'};
		int gg=0;
		if(suma<0 || suma>1)
			return 'z';
		for(gg=0;gg<24;gg++)
			if(Vi[gg]<suma && suma<Vf[gg]){
				return letra[gg];
		}
}
```

Prueba.cpp

```
#include "prueba_2.h"
#include "ui_prueba_2.h"
#include <QPixmap>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <qwidget.h>
#include<QString>
#include<QImage>
#include<QDir>
class QPushButton;

class QTextEdit;
class QLineEdit;

prueba_2::prueba_2(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::prueba_2){
   QString imagefilename;
            r=new PgmRotate();
            r2=new PgmRotate();
            h=new Haar2D();
            P=new Pesos();
    ui->setupUi(this);
}

prueba_2::~prueba_2()
{
    delete ui;
}

void prueba_2::changeEvent(QEvent *e){
    QMainWindow::changeEvent(e);
    switch (e->type()) {
            case QEvent::LanguageChange:
                ui->retranslateUi(this);
                break;
             default:
              break;
    }

}
void prueba_2::algo(){
    connect(ui->salir, SIGNAL(clicked()), qApp, SLOT(quit()));//salir
    connect(ui->cargar, SIGNAL(clicked()), this, SLOT(algo_2()));//limpia un qlabel
}
void prueba_2::algo_2(){
```

```
    QString imagefilename = QFileDialog::getOpenFileName( this,"Image to
open",QDir::currentPath(),"Images (*.pgm *.png *.xpm *.jpg)");//para abrir una imagen
    QImage   QImagen;
    QImagen.load ( imagefilename);
    ui->label_3->setPixmap(QPixmap::fromImage(QImagen));
    char alj[128];
  QString hh = imagefilename;
    QByteArray ba = hh.toLatin1();
    const char *c_str2 = ba.data();
 const char * fileName;
 char d;
     fileName=hh.toAscii();
   h->parseargs(fileName);
   d=P->leer();
   algo_5(d);
}
void prueba_2::algo_3(){
    ui->label_2->QLabel::setPixmap(QPixmap("x.pgm"));
    ui->label->QLabel::setPixmap(QPixmap("/root/Escritorio/listo/x.pgm"));
    QPixmap pixmap=QPixmap((tr(":/w.pgm")));
}
void prueba_2::algo_4(){//para mostrar un msg
    QMessageBox::information(
        this,
        tr("Application Name"),
        tr("An information message.") );
}
void prueba_2::algo_5(char d){
    QImage   QImagen2;
    if(d=='a')
        QImagen2.load ("a.pgm");
    if(d=='b')
        QImagen2.load ("b.pgm");
    if(d=='c')
        QImagen2.load ("c.pgm");
    if(d=='d')
        QImagen2.load ("d.pgm");
    if(d=='e')
        QImagen2.load ("e.pgm");
    if(d=='f')
        QImagen2.load ("f.pgm");
    if(d=='g')
        QImagen2.load ("g.pgm");
    if(d=='h')
        QImagen2.load ("h.pgm");
    if(d=='i')
        QImagen2.load ("i.pgm");
    if(d=='k')
        QImagen2.load ("k.pgm");
    if(d=='l')
        QImagen2.load ("l.pgm");
```

```
   if(d=='m')
      QImagen2.load ("m.pgm");
   if(d=='n')
      QImagen2.load ("n.pgm");
   if(d=='o')
      QImagen2.load ("o.pgm");
   if(d=='p')
      QImagen2.load ("p.pgm");
   if(d=='q')
      QImagen2.load ("q.pgm");
   if(d=='r')
      QImagen2.load ("r.pgm");
   if(d=='s')
      QImagen2.load ("s.pgm");
   if(d=='t')
      QImagen2.load ("t.pgm");
   if(d=='u')
      QImagen2.load ("u.pgm");
   if(d=='v')
      QImagen2.load ("v.pgm");
   if(d=='w')
      QImagen2.load ("w.pgm");
   if(d=='x')
      QImagen2.load ("x.pgm");
   if(d=='y')
      QImagen2.load ("y.pgm");
    if(d=='z')
           {          QMessageBox::warning(
                 this,
           tr("Warning"),
           tr("No reconocida.") );
           }
   ui->label_2->setPixmap(QPixmap::fromImage(QImagen2));

}
void prueba_2::algo_6(QString imagefilename){

}
```

Apéndice - Código fuente

Bibliografía

[1] World Federation of the Deaf.
www.wfdeaf.org/.
Consultado 20 Junio se 2010

[2]Edgar Ordoñez Ortíz. Ingeniero Electrónico en Computación. Agosto 2003. Egresado de la Escuela. 2006. Escuela Superior Politécnica de Chimborazo.
http://ieee.uach.cl/downloads/ingelectra2006/10_Ordonez.pdf.
Consultada 12 de Junio 2009

[3] Laura Jeanine Razo Gil.
http://docencia.dgsca.unam.mx/computo50/repositorio/jeaninePONENCIA001.pdf
Consultada 20 de Julio 2009

[4]José L. Illera. " DITS un programa informatico para el aprendizaje del códigodactológico'
http://dialnet.unirioja.es/servlet/fichero_articulo?codigo=2941579.pdf
Consultada 16 de Junio 2009.

[5]Autor: Fausto Mario Díaz Cabrera. Alumno de la UAM Azcapotzalco. Otoño 2008. Consultada 14 de Junio 2009. Proyecto terminal. Lic. En Computación. División CBI.

[6] Microsoft Windows Corporation
http://www.microsoft.com/

[7] Walker J.,"A primer on wavelets and their scientific applications". 2nd Edition Chapman Hall CRC,Taylor Francis Group,2008.
Consultada 7 de Junio de 2010

[8] Haykin, S., "Neuronal Networks a comprehensive foundation ", 2nd Edition ,Prentice Hall,1999.
Consultada 7 de Junio de 2010

[9] Data engine,
http://www.dataengine.de/.
Consultado en Junio de 2010

[10] QT de Nokia Marca registrada
http://qt.nokia.com/
Consultada 10 Abril 2010

[11] Digital Image Processing, Rafael C. Gonzalez, Richard E Woods., Third Edition, Pearson Prentice Hall
Consultado en 20 Junio 2010

[12] Códigos fuente PgmRotate.cpp, Haar2D.cpp autor: Disponible mediante solicitud con el autor, Dr. Oscar Herrera Alcántara.