Programación Matemática y Software (2018) 10(1): 44-56. ISSN: 2007-3283

Reconocimiento de caracteres mediante OCR (Optical Character Recognition)

Recognition of characters through OCR (Optical Character Recognition)

Gustavo Medina Ángel [1], Yessica Yazmin Calderón Segura [1], Gennadiy Burlak [1]
[1] Centro de Investigación en Ingeniería y Ciencias Aplicadas,
Universidad Autónoma del Estado de Morelos.
Avenida Universidad 1001, Colonia Chamilpa, Cuernavaca, Morelos, México, C.P. 62209.
gustavo.isc@hotmail.com, ycalderons@uaem.mx, gburlak@uaem.mx

PALABRAS CLAVE:

pixeles, matriz, iteración, normalización.

RESUMEN

En este trabajo, se implementaron técnicas para el reconocimiento digital de caracteres, utilizando la técnica OCR (Reconocimiento Óptico de Caracteres), que implementa métodos como binarización, etiquetado, esqueletización y proyección de trazas en las imágenes, para reconocer caracteres que optimizan los procesos en la abstracción y digitalización de libros, revistas u otras fuentes de información que pueden digitalizarse y luego manipularse en su formato digital.

KEYWORDS:

pixels, matrix, iteration, standardization.

ABSTRACT

In this work, techniques for the digital recognition of characters were implemented, using the technique OCR (Optical Character Recognition), which implements methods such as binarization, labeling, skeletonization and projection of traces in the images, in order to recognize characters that optimize the processes in the abstraction and digitalization of books, magazines or other sources of information that can be digitized and then manipulated in their digital format.

Recibido: 29 de julio del 2017 • Aceptado: 5 de noviembre del 2017 • Publicado en línea: 28 de febrero del 2018

1. Introducción

Actualmente existen muchas técnicas para el reconocimiento de caracteres como lo son el uso de redes neuronales y la correlación entre objetos [1], identificando los bordes mediante la transformada de Fourier, sin embargo la OCR ha demostrado tener excelentes resultados en cuanto al reconocimiento de caracteres.

La OCR, es una técnica de reconocimiento de imágenes, que involucran técnicas como lo son la binarización, segmentación, normalización, esqueletización y proyecciones.

En este proceso la imagen es leída, y guardada en una matriz o arreglo bidimensional, para posteriormente aplicarle filtros [2].

Los procedimientos a seguir para la OCR, se divide en las siguientes secciones: La seccion uno, presenta las tecnicas de la OCR. La seccion dos muestra las diferentes formas de obtener una imagen para su posterior reconocimiento. En la seccion tres ses convierte la imagen digital a una matriz de ceros y unos. En la seccion cuatro se segementa y etiqueta la matriz con el algoritmo Hoshen-Kopelman. En la seccion cinco se separan los caracteres y se identifican sus limites. En la seccion seis se aplica la normalizacion de la imagen. En la seccion siete, se adelgaza la imagen para su posterior comparacion. En la seccion ocho se realizan las proyecciones para realizar la comparacion con los caracteres pratrones. En la seccion nueve, se realizan las pruebas aplicando la OCR en una imagen muestra y se muestan los resultados. En la seccion diez se muestran las conclusiones y en la seccion once se presentan las referencias bibliograficas.

2. Adquisición de la imagen

Es el proceso por el cual la imagen es obtenida, mediante cámara, impresión de pantalla, escaneo de documentos, foto tomada por cámara web, etc.

Un aspecto que es importante considerar, es que de acuerdo a la calidad de imagen que se adquiera, corresponderá el buen o mal reconocimiento de las letras.

3. Binarización

La técnica de binarización es el proceso en la cual el pixel de una imagen es separado en solo dos valores que pueden ser 0 o 255. Donde el valor 0 es el color negro y el valor 255 será un color blanco [3].

3.1 Umbral de la binarización

Como mencionamos anteriormente, una imagen puede tomar un valor que va desde 0 a 255 [4], pero quien decide que este pixel tome un valor de 0 o 255, es la variable llamada: umbral de binarización [5], y a partir de esa variable, todos los valores que sean menores o iguales a ese umbral se convertirán en 0, los que sean mayores se convertirán en 1.

El siguiente ejemplo muestra la binarización de una imagen con un umbral de 120.

En la Figura 1, se muestra se muestra una imagen escaneada sin binarizar.

Después de aplicar el proceso de binarización a la imagen anterior, solo serán visibles los pixeles más oscuros [6], es decir todos aquellos con un umbral más alto que 120, tal como se muestra en la figura 2.

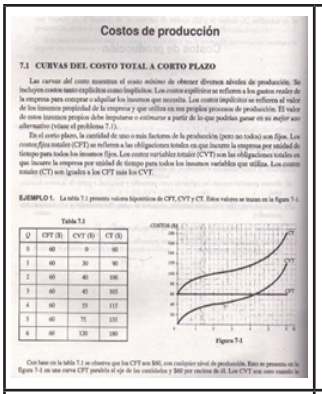


Figura 1-Imagen original sin binarizar.
Fuente: Elaboración propia

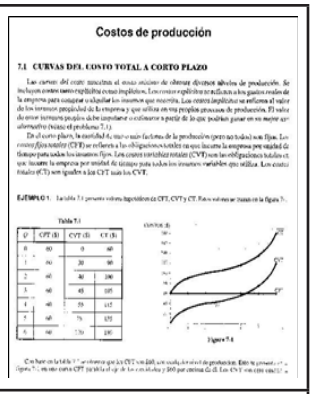


Figura 2-Imagen binarizada.
Fuente: Elaboración propia

Como se puede observar en la segunda imagen, todos aquellos detalles que aparecían en la primera imagen fueron eliminados, sombras, puntos y degradados.

Este método puede ser un arma de doble filo, ya que si el umbral es muy alto la imagen puede ser demasiado clara y perderse información importante, algunos caracteres (sobre todo los delgados) pueden desaparecer [7]. En caso contrario, si el umbral es demasiado bajo, la mayor parte o casi toda la imagen será de un aspectos oscuro o negro en su totalidad [8].

3.2 Algoritmo de Binarización.

Por ejemplo teniendo:

```
Byte Imagen[] ← Leer Imagen
Entero Umbral← 127, i, j
Entero M[]←Imagen[]
Para i ← 0 hasta i<NumeroFilas hacer
Para j ← 0 hasta j <=NumeroColumnas hacer

Si M[j,i]<=umbral
M[j,i]← 0
Si no
M[j,i]←255
Fin si
Fin para j
Fin para i
```

Se carga la imagen en el arreglo "Imagen", se declara el umbral que será el valor con el cual se condiciona los valores de la matriz para que estos tomen un valor de 0 ó 255.

Todos aquellos valores de la matriz con algún valor menor a 127, se convertirán en ceros, en caso contrario se le asignara un valor de 255.

En Figura 3, se muestra el resultado del algoritmo de binarización de una matriz de 7 filas por 7 columnas.

255	255	0	0	0	0	255
0	255	0	255	0	0	0
0	255	255	255	0	255	255
0	0	0	0	0	0	255
0	255	0	0	255	0	0
0	255	0	0	255	0	0
255	0	0	0	255	0	0

Figura 3.-Muestra valores de binarización.
Fuente: Elaboración propia

Posteriormente, a esta matriz la convertimos a una matriz binaria de 1's y 0's.

Para todos aquellos valores donde el valor sea 255, le asignaremos un 1.

```
Entero i,j
Para i ← 0 hasta  i<NumeroFilas hacer
Para j ← 0 hasta j <=NumeroColumnas hacer
Si M[j,i]==255 entones
M[j,i]← 1;
Fin si
Fin Para i
Fin Para j
```

En la figura 4, se muestra el resultado de la matriz resultante en donde los valores 255 pasaron a tomar el valor de 1.

1	1	0	0	0	0	1
0	1	0	1	0	0	0
0	1	1	1	0	1	1
0	0	0	0	0	0	1
0	1	0	0	1	0	0
0	1	0	0	1	0	0
1	0	0	0	1	0	0

Figura 4.-Muestra los valores de binarización con 0's y 1's.
Fuente: Elaboración propia

4. Segmentación

También llamado etiquetado de la matriz, es el proceso en el cual todos aquellos pixeles en la matriz forman un grupo de 1's separado de otros grupos [9], a este grupo comúnmente son llamados también clúster [10].

4.1 Algoritmo Hoshen-Kopelman (HK)

El algoritmo HK consiste en escanear a toda la matriz solamente para aquellos valores de la matriz que tengan el valor de uno en su celda. Para éstos se analizan los vecinos cercanos que se encuentren alrededor de una celda.

En este algoritmo se analiza el valor de la izquierda y el valor superior de la celda que se está analizando, si ambos son 0, quiere decir que no está unido a un grupo o clúster de 1's.

Si el valor de la izquierda es 1 y el de arriba es 0 entonces, se etiqueta con el último valor de etiqueta que existió, si los dos son 1, se etiqueta con el ultimo valor de le etiqueta que ocurrió [11].

Algoritmo:

```
Entero largest_label ← 0, x, y
Para  x← 0 hasta numero_columnass hacer
Para y ←0 hasta numero_filas hacer
Si occupied[x,y]==1 entonces
left ← occupied[x-1,y]
above ← occupied[x,y-1]
Fin si
Si  left == 0 y above == 0 entonces
largest_label ← largest_label + 1
label[x,y] ← largest_label
Sino
Si left != 0 y bove == 0 entonces
label[x,y] ← find(left)
Sino
Si left == 0 and above != 0 then entonces
label[x,y] ← find(above)
sino entonces
union(left,above)
label[x,y] ← find(left)
Fin si
Fin si
Fin si
Fin para x
Fin  para y
```

Se declaran la variable entera largest_label, donde se acumularán los valores desde 0 hasta un valor máximo encontrado, para etiquetar un carácter.

Se declaran también x e y para los ciclos de repetición.

4.2 Métodos Auxiliares del Algoritmo Hoshen-Kopelman (HK)

El método find, etiqueta a todos los valores de la matriz que tengan 1 arriba de la celda o a la izquierda de la matriz. Se crea una matriz llamada labels para ir guardando el conteo de agrupaciones y a la vez para devolver el valor de la etiqueta para cada celda encontrada de la matriz.

```
Entero labels[1000]
Función find(x : entero) : Entero
Mientras labels[x] != x hacer
   x <- labels[x]
Fin Mientras
 retornar x
Fin Función find
```

El método unión, funciona como un segundo método de etiquetado, y este invoca al método fin para etiquetarlo y para guardar el valor de la etiqueta en la matriz de label.

```
Procedimiento union(x: entero, y: entero)
 labels[find(x)] = find(y)
Fin Procedimiento unión
```

En la figura 5, se muestra el resultado de aplicar el algoritmo Hoshen-Kopelman donde se puede apreciar las agrupaciones.

Para cada grupo existe una etiqueta numérica diferente.

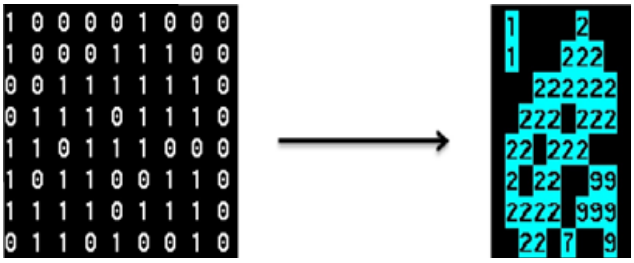


Figura 5.-Matriz binaria con la aplicación del Algoritmo HK.

Fuente: Elaboración propia.

Prueba 1 del Algoritmo HK con caracteres:

En la Figura 6, se muestran los caracteres que se quieren etiquetar y su resultado después de la segmentación o etiquetado, en este ejemplo se etiquetan los caracteres 'a' y 'e'.

Se pueden apreciar los caracteres etiquetados con los valores 1 y 4 para el carácter 'a' y, 2 y 5 para el carácter 'e'.

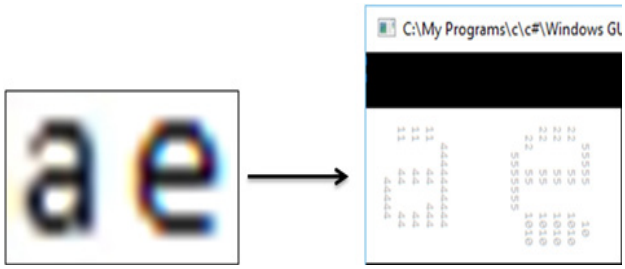


Figura 6 -Imagen que es etiquetada con una resolución muy pequeña de 19x11 pixeles

Fuente: Elaboración Propia

Prueba 2 con caracteres más grandes:

Cuando las imágenes tienen mejor resolución, los resultados para el etiquetado son más precisos.

En la figura 7, se muestra el resultado de etiquetar un texto con la palabra "hola", podemos observar que se asignó una etiqueta para cada carácter; 1 para la letra 'h', 4 para la letra 'o', 2 para letra 'l' y 5 para la letra 'a',

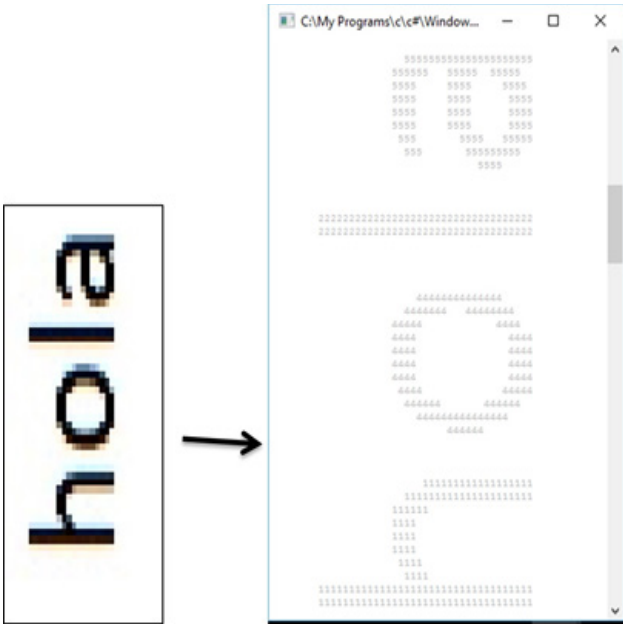


Figura 7.-Imagen que muestra el etiquetado de los valores de una imagen vertical de otro texto mayor resolución, de 52 x 27 pixeles.

Fuente: Elaboración Propia

El proceso para etiquetar una imagen es el proceso más caro en el procesamiento de reconocimiento de patrones [12], y en muchos casos, las diferentes técnicas para etiquetar una imagen son eficaces pero con poca calidad o eficientes pero con más consumo de memoria.

5. Obtención de la información de la Imagen

Una vez que se etiquetó al carácter con un número se tiene que obtener la información para independizarlo de todas las demás etiquetas, para esto debemos de analizar todas las coincidencias y debemos encontrar los valores con el valor de la etiqueta en sus 4 punto x1, y1 (punto de inicio del grupo con el mismo número) y x2, y2 (Puntos finales del grupo con el mismo número) tal como se muestra en la Figura 8 [13].

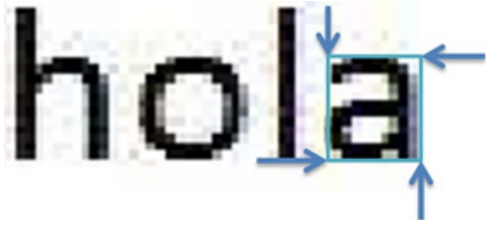


Figura 8.-Imagen que muestra los límites (coordenadas) de un carácter

Fuente: Elaboración Propia

5.1.-Algoritmos para la obtención de la información

Se desean encontrar x1,y1,x2 e y2, para esto debemos de saber primero cual es el número de etiqueta, posteriormente hacer 4 iteraciones dobles para cada punto del clúster y así poder obtener x1,y1,x2 e y2.

```
Se declaran:
Entero i,j
Para encontrar y1:
Etiqueta<-n
Para i<-0 hasta i< filasMatriz hacer
Para j<-0 hasta j< ColumnasMatriz hacer
Si matriz[i,j]==Etiqueta y matriz[i,j]==matriz[i,j+1] hacer
y1<-i
Romper
Fin si
Fin Para j
Fin Para i
```

Se realizan los ciclos anidados para i y j y si el valor de la posición de la matriz[i,j] es igual al valor de etiqueta y la siguiente posición debajo (matriz[i,j+1]) es igual al valor de la posición actual(matriz[i,j]), entonces el valor de y1 será el valor de la iteración i.

La variable N puede tomar un valor desde 0 hasta N(Numero de caracteres encontrados y etiquetados con un valor numérico).

Para encontrar x1, se tienen los ciclos anidados con las variables j e i, esta vez se comparan la posición actual matriz[i,j] y si tiene el valor de la etiqueta actual y si también el valor de esta posición es igual a la posición de delante de la matriz(matriz[i+1,j]), entonces el valor de x será j.

Para encontrar y2 se tiene dos ciclos anidados con las variables j e i. Si el valor de la posición actual de la matriz (matriz[i,j]) es igual al valor de la etiqueta que se está analizando y si el valor actual también es igual a el valor de la posición superior del valor que se está analizando, entonces el valor de y2 será el valor de la iteración de la variable i.

Para encontrar el valor de x2 se realizan un anidamiento de dos ciclos con las variables j e i, si la posición actual(matriz[i,j) es igual al valor de la etiqueta que se analiza y si también el valor de la posición actual(-matriz[i,j) es igual al valor de la posición de la siguiente columna(matriz[i-1,j]), entonces el valor de x2 será el valor de la iteración j.

Conociendo los valores de x1, y1, x2 e y2, se crea una

matriz para la etiqueta a extraer y se pasa la información de la matriz que contiene las etiquetas a la nueva Matriz M.

```
Entero Filas<-x2-x1
Entero Columnas<-y2-y1
Entero M[Filas, Columnas],i,j
Para i<-0 hasta i<Filas hacer
Para j<-0 hasta j<Columnas hacer
M[i,j]<-matriz[i+x1,j+y1] hacer
Fin Para j
Fin Para i
```

Para construir la matriz para el carácter, es necesario saber el tamaño de la matriz para guardar la etiqueta del carácter, y para saber el número de filas lo hacemos restando x2-x1 (Filas=x2-x1) mientras que el número de filas se obtiene restando y2-y1 (Columnas=y2-y1).

Posteriormente utilizamos dos ciclos anidados uno con la variable i y otro con la variable j para llenar un arreglo M, donde se pasan los valores del carácter o etiqueta de la matriz.

6. Normalización

La normalización consiste en tener el mismo número de pixeles en una imagen, es decir una imagen que es nxm, se convertirá en una imagen nxn (cuadrada).

Para normalizar la matriz aprovechamos un método de C#, que normaliza la imagen a partir de su lado más grande, y la deja de un solo tamaño, el método es: InterpolationMode.HighQualityBicubic [14].

El pseudocódigo que se utilizó para normalizar fue el siguiente:

```
Imagen imgToResize <-LeerImagen
Entero ancho<-ObtenerAnchoImagen(imgToResize)
Entero altura<-ObtenerAltoImagen(imgToResize)
si ancho>altura
altura=ancho
si no
ancho<-altura
Fin si
MapaBits b<-construir MapaBits(ancho, altura)
Grafico g<-Grafico.Imagen(Imagen)b
d.DibujarImagen(imgToResize ,0,0,ancho,altura)
```

Se lee la imagen y se guarda en una variable de tipo Imagen; imgToResize. Se obtiene el ancho y la altura de la imagen y se guardan en las variables enteras ancho y altura. Se comparan los valores, si el valor más grande es la de la altura, el tamaño de ancho pasa a tener el valor de altura y si el valor de más grande es ancho, el tamaño de altura pasa a tener el valor de ancho.

Se crea un mapa de bits con la variable b que será construido a partir del ancho y altura obtenidos anteriormente (MapaBits b=construir MapaBits(ancho, altura)).

Para dibujar la imagen se crea un valor de tipo Grafico (Grafico g<-Grafico.Imagen(Imagen)b), que tendrá el tamaño de del mapa de bits b.

Por último se dibuja una nueva imagen con los nuevos valores de ancho y alto (g.DibujarImagen(imgToResize ,0,0,ancho,altura)).

En la Figura 9, se muestra el resultado de la normalización y se puede observar que la imagen de prueba con el carácter 'a' cambio de una resolución de 81 x 100 pixeles a 100 x 100 pixeles.

Imagen antes de normalizar:	*Imagen después de normalizar:*
	

Figura 9.-Imagen que muestra el antes y el después de un carácter al ser normalizado

Fuente: Elaboración propia.

Se normalizó la imagen tomando el tamaño del lado más grande.

7. Adelgazamiento de los componentes

Una vez separado por segmentos los componentes, se aplica un proceso de adelgazamiento para cada uno de ellos [15], que consiste en ir borrando los puntos sobrantes del contorno de cada componte, de modo que conserve su tipología [16].

7.1. Algoritmo de Esqueletización Pavlidis

Este algoritmo consiste en adelgazar una matriz de entrada, e ir comparando con una pequeña matriz de plantilla, y esta debe ser comparada con los 4 vecinos de cada pixel de la matriz de la imagen [17].

Un pixel es esqueletal si alguno de ellos presenta

alguna similitud con alguno de estas plantillas de 3x3, que se muestran en la figura 10.

X	X	X
0	1	0
Y	Y	Y

X	X	X
0	1	X
1	0	X

X	X	Z
0	1	D
Y	Y	Z

Figura 10.-Plantillas con las que se comparan cada valor o pixel para cada uno de los caracteres.

Fuente: Elaboración Propia

Algoritmo:

```
Entero remain <- 1
Entero skel <- 0;
Entero nf <-FilasMatriz
Entero nc <- ColumnasMatriz
Entero d <- 1
Entero i,j,k
mientras  remain==1 hacer
    remain <- 0;
    Para k<-1 hasta k<=4 hacer
       Para i<-1+d hasta i<=nf-d hacer
          Para j<-1+d hasta j<=nc-d hacer
             Si matriz[i,j-1]==0 hacer
                skel <- 0
si matriz[i,j]== borde1[i-1,j]   o matriz[i,j]== borde1[i+1,j]   o
matriz[i,j]== borde1[i,j-11]   o matriz[i,j]== borde1[i,j+1]
matriz[i,j]== borde2[i-1,j]   o matriz[i,j]== borde2[i+1,j]   o
matriz[i,j]== borde2[i,j-11]   o matriz[i,j]== borde2[i,j+1]
matriz[i,j]== borde3[i-1,j]   o matriz[i,j]== borde3[i+1,j]   o
matriz[i,j]== borde3[i,j-11]   o matriz[i,j]== borde3[i,j+1]
si skel== 1 hacer
matriz[i,j] <- 2
Si no    matriz[i,j] <- 3
         remain <- 1
                Fin si
       Fin Para j
Fin Para i
Para ii<-1+d hasta ii<=nf-d hacer
          Para jj<-1+d hasta jj<=nc-d hacer
             Si matriz[i,j]==3 hacer matriz[i,j]<-0
Fin si
Fin Para j
Fin Para i
Fin Para k
Fin mientras
```

Las variables remain indicará si existen pixeles esqueletales a partir de la comparación con sus vecinos.

Skel es una variable entera que nos indica si el pixel comparado y sus vecinos son iguales a alguna de las tres platillas.

Las variables nf y nc indican el número de filas y columnas de la matriz binarizada, la variable "d", se declara para dejar un margen para la primera fila y la última fila, así como para la primera columna y última columna de la matriz, para que los filtros comiencen a compararse una posición después de la imagen hasta una posición antes de la imagen.

Mientras haya alguna coincidencia (remain==1) con alguno de los tres filtros, se analizara la matriz.

La variable skel es parecida a remain, solo que este actúa de forma interna en los ciclos.

El ciclo para k, analiza las 4 posiciones de los índices vecinos de un determinado punto en la matriz.

Mientras los ciclos con los índices i y j analizan todas las posiciones del arreglo excepto: el primer renglón, el último renglón, la primera columna y la última columna de la matriz.

Si alguna de los cuatro vecinos se parece a los de algún filtro de la matriz s, entonces skel es igual a 1, y por lo tanto se asigna a la matriz un 2 (matriz[i,j] = 2), de lo contrario se le asigna un 3(matriz[i,j] = 3) y a la variable remain el valor de 1 que indica que el ciclo se debe repetir nuevamente.

Los ciclos para ii y jj recorren la matriz para encontrar todos los "3" que existan (Si matriz[i,j]==3 hacer) y los convierten en ceros y los reasignan en la misma posición(matriz[i,j]==0).

En la Figura 11, se muestra el resultado de aplicar el algoritmo de esqueletización a una matriz que contiene el carácter 'a'. Se observa como la imagen ´original pasa a ser solo el esqueleto formado con los pixeles necesarios para su reconocimiento [17].

Figura 11.-Imagen que muestra la figura esqueletal del carácter "a"

Fuente: Elaboración propia.

8. Comparación con caracteres patrones

En esta etapa, se comparan los caracteres obtenidos a partir del adelgazamiento con los caracteres teóricos o caracteres patrones guardados en archivos y previamente cargados en el programa en matrices independientes para cada una [18].

8.1. Modelo de proyección horizontal y vertical de trazas

En este modelo se crean dos vectores, un vector que tendrá los valores de la proyección vertical y otro que tendrá los valores de la proyección horizontal.

Un vector de trazas es un arreglo unidimensional con tamaño f (filas si es horizontal) y c (columnas si es vertical).

El vector de trazas guardará el número coincidencias en donde haya trozos de un carácter, y acumulará estas coincidencias para después compararlo con las del carácter patrón [19].

En las Figuras 12 y 13, se muestran las proyecciones vertical y horizontal, en donde se visualizan las coincidencias por filas y columnas almacenadas en el vector de trazas.

Figura 12.-Proyección de trazas horizontal
Fuente: Elaboración propia.

Figura 13.-Proyección de trazas vertical
Fuente: Elaboración propia.

8.2. Algoritmo de proyección vertical y horizontal

```
Entero vectorHorizontal[filas]
Entero vectorVertical[columnas]
Entero trazos<-0
```

Se declaran los arreglos enteros vectorHorizontal y vectorVertical para guardar el número de trazas que se encuentre en una proyección y una variable entera para contar cuantas trazas se encontraron al recorrer la matriz ya sea horizontalmente o verticalmente.

```
Para i<-0 hasta i<=filas hacer
Para j<-0 hasta j<=columnas-1 hacer
Si matriz[i,j]!=matriz[i,j+1] hacer Trazos<-trazos+1
Fin si
Trazos<-trazos /2
vectorHorizontal[i]<-trazos
fin para i
fin para j
```

Para las trazas horizontales la variable j lleva las filas de nuestra matriz, si la posición de la matriz i, j es diferente a la de la siguiente posición de la matriz i,j+1 entonces se dice que comenzó un pixel en 1, y cuando vuelva a encontrarse una posición diferente, la traza termina. Cada vez que se encuentre una diferencia de pixel se incrementara la variable de trazos, se divide entre dos el valor de trazos ya que solo se quiere saber el número de trazas y no el número de veces que comienza y termina una traza.

Por último se guarda el valor encontrado en la matriz de trazos Horizontal.

```
Para i<-0 hasta i<=columnas-1 hacer
Para j<-0 hasta j<=filas hacer
Si matriz[i,j]!=matriz[i+1,j] hacer
Trazos<-trazos+1
fin Si
vectorVertical[i]<-trazos
fin para i
fin para j
```

Para las trazas verticales, se sigue el mismo comportamiento que la de las trazas horizontales, solo que este dependerá de la variable i, que el lleva el control de las columnas de la matriz.

El valor del incremento de las trazos en matriz[i,j]!=-matriz[i+1,j], se guardará en el vector de vector de trazas vectorVertical.

8.3. Comparación de proyecciones

En este algoritmo, se comparan las proyecciones de los caracteres a reconocer con las proyecciones de los caracteres patrón [20].

```
Entero  proyeccionMenor<-0
Entero  ProyecciónAnterior<-0
Entero  por<-0
Entero  pver<-0
Entero  NumeroCaracter<-0
Entero  NCatacter<-0
Para  i<-0  hasta i< arregloCaracteres entonces
Cambio (arregloCaracteres[i])
Caso 1  hacer
ProyecciónAnterior<- proyeccionMenor
phor <- proyeccionesHor [i]- vectorHorizontal[i]
pver<- proyeccionesVer[i]- vectorVertical[i]
proyeccionMenor<- por+ pver
Si phor ==0 y pver==0 hacer
ProyecciónAnterior <-proyeccionMenor


Sino
proyeccionMenor<- ProyecciónAnterior
Ncarácter<-i
Car<- 'a'
fin si
romper
.
.
.
Caso N
ProyecciónAnterior<- proyeccionMenor
phor <- proyeccionesHor [i]- vectorHorizontal[i]
pver<- proyeccionesVer[i]- vectorVertical[i]
proyeccionMenor<- por+ pver
Si phor ==0 y pver==0 hacer
ProyecciónAnterior <-proyeccionMenor
Sino
proyeccionMenor <- ProyecciónAnterior
Ncarácter<-i
Car<- 'n'
fin si
romper
Fin cambio
Imprimir car
Fin para i
```

Se toman todas las proyecciones para el arreglo de caracteres que contiene las etiquetas declarados

en el programa, y se compara las proyecciones para cada carácter con las proyecciones de la etiqueta de la imagen leída, si el valor de ambas proyecciones es igual a 0, entonces se encontró el carácter con las mismas proyecciones.

ProyeccionesHor y proyeccionesVer son los valores de las proyecciones declarados en el programa con los valores de las proyecciones para cada caso (carácter), sirven para recorrer todos los casos, hasta que se encuentre el carácter correcto.

Se asigna el valor del carácter (Car<- 'a') y al final de todos los casos se imprimirá el carácter.

9. Pruebas y resultados

A continuación se muestra un ejemplo en donde se muestra una imagen seleccionada para su conversión a texto en un programa que se desarrolló en Microsoft Visual C# [21].

Se debe seleccionar el texto al cual se quiere obtener la imagen, como se muestra en la figura 14.

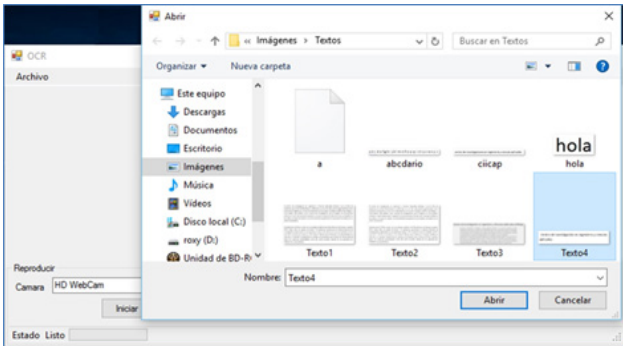


Figura 14 Imagen que muestra selección del archivo mediante

Fuente: Elaboración propia.

Se debe seleccionar el texto al cual se quiere obtener la imagen.

La imagen se visualizará del lado izquierdo y del lado derecho aparecerá la extracción de caracteres que se obtuvo de este texto como se muestra en la figura 15.

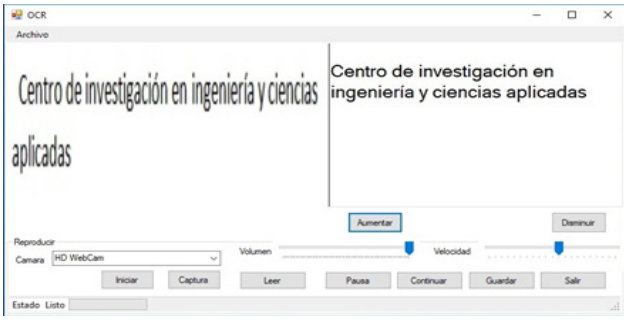


Figura 15-Imagen que muestra el resultado del reconocimiento del archivo seleccionado.

Fuente: Elaboración propia.

La resolución del texto leído es de 379x71 pixeles, y el resultado fue el siguiente: "Centro de investigación en ingeniería y ciencias aplicadas", con una conversión del 100%.

9.1. Resolución mínima para reconocimiento con OCR

Se realizaron pruebas con una muestra tomada de una imagen con la palabra "hola" con una resolución de 56x31 pixeles y se decremento la calidad de la resolución hasta 51x26 pixeles en donde observamos que las imágenes mayores o iguales a esta resolución aún son reconocidas.

Como se muestra en la tabla 1, con una resolución de 50x25 pixeles, los resultados de esta prueba ya no son de buena calidad y por lo tanto no son reconocidos por la OCR.

Tabla 1-Muestra los resultados del decremento de resolución de una imagen y el resultado de su reconocimiento mediante OCR.

Fuente: Elaboración propia.

No. Prueba	Resolución	Resultado
1	56x31 pixeles	hola
2	55x30 pixeles	hola
3	54x29 pixeles	hola
4	53x28 pixeles	hola
5	52x27 pixeles	hola
6	51x26 pixeles	hola
7	50x25 pixeles	No se reconoce el texto
8	49x24 pixeles	No se reconoce el texto
9	48x23 pixeles	No se reconoce el texto
10	47x22 pixeles	No se reconoce el texto

Pudimos observar en esta prueba que la resolución mínima para el caso de la palabra “hola” es de 51x26 pixeles.

Cabe señalar que el punto a decrementar de la imagen es en su lado vertical, ya que al leer un texto generalmente este lado es el de menor resolución y la altura es igual para todos los caracteres en la imagen, mientras que el lado horizontal puede crecer dependiendo de la cantidad de texto o letras en la imagen.

9.2. Imagen reconocida con Webcam.

Se obtiene también la opción de cámara, en el cual nosotros podemos capturar una imagen en vivo [22] y reconocer los caracteres en ella, como se muestra en la figura 16.

Cabe señalar, que los resultados obtenidos en esta prueba no fueron satisfactorios, ya que el texto que se esperaba obtener no era lo suficiente mente bueno.

Esto debido a la calidad de la cámara, sombras, movimientos que pueden distorsionar la imagen, así como otros objetos ajenos en el fondo de la imagen a reconocer [23].

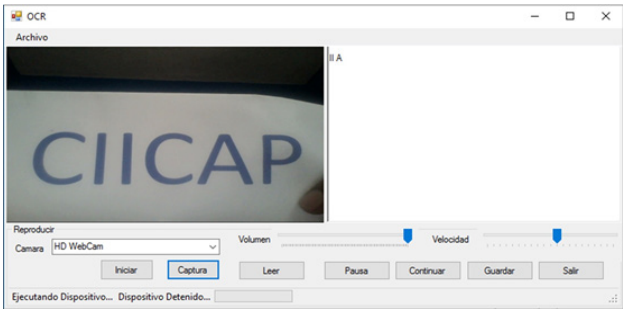


Figura 16.-Imagen que muestra el reconocimiento de una imagen obtenida de la cámara web

Fuente: Elaboración propia.

10. Conclusiones

Al realizar las pruebas se observó que entre menos resolución tenga la imagen, los resultados serán menos satisfactorios y el OCR puede no reconocer algunos caracteres o son remplazados por otros caracteres equivocados.

Se determinó que la resolución mínima para el lado vertical de la imagen debe ser de 26 pixeles para que la OCR pueda reconocerla.

Cuando las imágenes son tomadas en tiempo real, muchos factores pueden influir, por ejemplo la luz, el movimiento y el ángulo de inclinación con la que se captura la imagen, es por ello que al realizar esta prueba los resultados en su mayoría fueron no favorables.

REFERENCIAS

[1] Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y. A methodology for feature selection using multiobjective genetic algorithms for handwritten digit string recognition. International Journal of Pattern Recognition and Artificial Intelligence. 2003,17(06), 903-929.

[2] Bazzi, I., Schwartz, R., Makhoul, J. An omnifont open-vocabulary OCR system for English and Arabic. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1999, 21(6), 495-504.

[3] Bortolozzi, F., Britto Jr, A.S., Oliveira, L.S., Morita M. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002, 24(11), 1438-1454.

[4] Ramírez-Ortegón, M.A., Tapia, E., Ramírez-Ramírez, L.L., Rojas,R., Cuevas, E. Transition pixel: A concept for binarization based on edge detection and gray-intensity histograms. Pattern Recognition. 2010, 43(4), 1233-1243.

[5] Ramírez-Ortegón, M.A., Ramírez-Ramírez, L.L., Märgner,V., Messaoud, I.B., Cuevas, E., Rojas, R. An analysis of the transition proportion for binarization in handwritten historical documents. Proceedings of 8th International Conference on Document Analysis and Recognition. 2014, 17(2), 139-160.

[6] Ramírez-Ortegón M.A., Märgner, Volker., Cuevas,E., Rojas, R. An optimization for binarization methods by removing binary artifacts. Pattern Recognition Letters. 2013, 34(11), 1299-1306.

[7] Čisar, P., Čisar, S. M., Subošić, D., Đikanović, P., & Đukanović, S. Optimization Algorithms in Function of Binary Character Recognition. Acta Polytechnica Hungarica. 2015, 12(7), 77-87.

[8] Yokobayashi, M., Wakahara,T. Binarization and Recognition of Degraded Characters Using a Maximum Separability Axis in Color Space and GAT Correlation. Pattern Recognition. 2006, 2, 885-888.

[9] Morita M., Sabourin, Robert., Bortolozzi, F., Suen, C.Y. Segmentation and recognition of handwritten dates: an HMM-MLP hybrid approach. Document Analysis and Recognition. 2003, 6(4), 248-262.

[10] Yokobayashi, M., Wakahara,T. Segmentation and recognition of characters in scene images using selective binarization in color space and GAT correlation. Document Analysis and Recognition. 2005, 1,167-171.

[11] Hoshen, J., R, Kopelman. Percolation and cluster distribution I: Cluster multiple labeling technique and critical concentration algorithm. Physical Review B. 1976 , 14(8), 3438-3445.

[12] Britto-Jr, A.S., Sabourin R., Bortolozzi F. The recognition of handwritten numeral strings using a two-stage HMM-based method. International Journal on Document Analysis and Recognition. 2003, 5(2-3), 2003.

[13] Zhong, Y., Karu, K., & Jain, A. K. Locating text in complex color images. Pattern recognition. Pattern recognition. 1995, 28(10), 1523-1535.

[14] Friston, K., Ashburner, J., Frith, C. D., Poline, J. B., Heather, J. D., & Frackowiak, R. S. Spatial registration and normalization of images. Human brain mapping. 2014, 3(3), 165-189.

[15] Mozaffari, S., Faez, K., Märgner, V., El-Abed, H. Lexicon reduction using dots for off-line Farsi/Arabic handwritten word recognition. Pattern Recognition Letters. 2008, 29(6), 724-734.

[16] Lam, L., Lee, S. W., & Suen, C. Y. Hinning methodologies-a comprehensive survey. EEE Transactions on pattern analysis and machine intelligence. 2011, 14(9), 869-885.

[17] Stentiford, F.W.M., & Mortimer, R.G. Some new heuristics for thinning binary handprinted characters for OCR. EEE Transactions on Systems, Man, and Cybernetics. 1983, SMC-13(1), 81-84.

[18] Watrous, L. E., & Wheeler, Q. D. The out-group comparison method of character analysis. Systematic Biology. 1981, 30(1), 1-11.

[19] Chen, J. L., & Lee, H. J. An efficient algorithm for form structure extraction using strip projection. Pattern recognition. 1998, 31(9), 1353-1368.

[20] Schneider, J. W., & Borlund, P. Matrix comparison, Part 1: Motivation and important issues for measuring the resemblance between proximity measures or ordination results. Journal of the Association for Information Science and Technology. 2007, 58(11), 1586-1595.

[21] S.F.J. Ceballos. MICROSOFT C#. LENGUAJE Y APLICACIONES. 2nd ed.: RA-MA EDITORIAL, 2007.

[22] Elagouni, K., Garcia, C., Mamalet, F., & Sébillot, P. Text recognition in multimedia documents: a study of two neural-based ocrs using and avoiding character segmentation. Journal on Document Analysis and Recognition. 2014, 17(1), 19-31.

[23] Li, H., Doermann, D., & Kia, O. Automatic text detection and tracking in digital video. IEEE transactions on image processing. 2000, 9(1), 147-156.

SEMBLANZA

Dr. Gennadiy Burlak

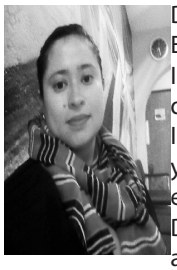
En 1975 estudió la licenciatura y maestría en la Universidad Nacional de Kiev (KNU), en la Facultad de Física y en el Departamento de Física Teórica. El Ph. D. (candidato en Ciencias físico-matemáticas) y el D. Sc. (doctor en Ciencias físico-matemáticas), los obtuvo también en la KNU en 1979 y 1988, respectivamente. Trabajó como catedrático del Departamento de Física Teórica. Actualmente, es Profesor-Investigador Titular "C" del Centro de Investigaciones en Ingeniería y Ciencias Aplicadas (CIICAp) de la Universidad Autónoma del Estado de Morelos (UAEM), desde 1998.

El Dr. Burlak es autor y coautor de cuatro libros y 140 artículos en revistas internacionales. Ha participado en 157 ponencias en congresos nacionales e internacionales. Bajo de su dirección han graduado: 3 tesis de doctorado y 8 tesis de maestría y licenciatura. Actualmente 4 tesis de doctorado en proceso bajo de su dirección.

Mtro. Gustavo Medina Ángel

En 2008 estudio la Ingeniería en Sistemas Computacionales egresado de Instituto Tecnológico de Zacatepec (ITZ), en el 2016 se tituló como Maestro en Ingeniería y Ciencias Aplicadas de la Universidad Autónoma del Estado de Morelos (AUEM), actualmente es profesor de la Facultad de Contaduría, Administración e Informática asi como en la escuela de Estudios Superiores de Mazatepec y es desarrollador de Software Independiente. Ha impartido talleres de programación avanzada en Java (2015), en el Centro de Educación Continua (CEC) del Instituto Politécnico Nacional (IPN), también ha impartido talleres de Programación en Arduino dentro de la UAEM y ha participado como asesor en línea para el curso de programación en Android para Docentes de Nivel Medio Superior, por medio de la SEP-eUAEM.

Actualmente dirige la tesis para la obtención de la licenciatura en informática de dos asesorados.



Dra. Yessica Yazmin Calderón Segura

En el 2009 obtuvo su título de Ingeniera en Informática en la universidad Politécnica del Estado de Morelos. Su maestra en Ingeniería y Ciencias Aplicadas en el 2011 y en el 2015 obtuvo el grado de Doctor en Ingeniería y Ciencias Aplicadas. La Dra. Yessica Yazmin Calderón Segura, actualmente trabaja como catedrática de la Facultad de Ciencias Químicas e Ingeniería, así mismo en la Facultad de Contaduría, Administración e Informática y se desempeña como ayudante de SNI III, escribiendo artículos de alto Impacto y participando en congresos Nacionales e Internacionales, con el Dr. Gennadiy Burlak del Centro de Investigación en Ingeniería y Ciencias Aplicadas. La Dra. Yessica Yazmin Calderón Segura, tiene amplios conocimientos en algoritmos, análisis de ciclos computacionales, estadística de datos, modelos de procesos de datos, nano estructuras, optimización y programación. La Dra. Yessica Yazmin Calderón Segura, termino sus estudios en el Centro de Investigación en Ingeniería y Ciencias Aplicadas con una tesis muy detallada y de excelente contenido. Con esta tesis la Doctora ha publicado 12 artículos en revistas indexadas internacionales y nacionales. Ha participado en 40 congresos Nacionales e Internacionales y cuenta con el Premio Estatal al Investigador.