DIGITALIZACIÓN Y RECONOCIMIENTO DE DOCUMENTOS MANUSCRITOS PARA LA PRESERVACIÓN DE PATRIMONIO CULTURAL

Prof. Ing. Marisa R. De Giusti[1]
A.C. Maria Marta Vila[2], A.C. Gonzalo Luján Villarreal[3]

Abstract: The handwritten manusctipt recognizing process belongs to the iniciatives which lean to cultural patrimony preservation shielded in Libraries and files where there exists a big wealth in documents and even handritten cards that accompany incunable books. This work is point to begin with a research and development proyect oriented to digitalization and recognition of manuscipt materials and the paper presented here discuss diferent algorithms used in the first stage dedicated to "noise-clean" of the image in order to improve it before the character recognition process begins. Since PrEBi – SeDiCI belong to a network of libraries that interchange digitalized documents by scanning, this document has brought up an extra use related to improvement of the images of interchange documents which presented common problems in its digitalization, such as: borders, impurity, not-centered texts, etc… Although it is not the final purpose of this research, it is still a very usefull skill within the framework of libraries consortium interchange.
In order to make the handwritten-text recognition and image digitalization process eficient, it must be preceded by a preprocessing stage of the image to be trated which includes thresholding, noise cleaning, thinning, base-line alignment and image segmentation, among others. Each one of these steps will allow to reduce the injurious variability when recognizing manuscripts texts (noise, random gray levels, slanted chacarters, ink level in different zones), and so increasing the probability of obtaining a suitable text recognition. In this paper, two image thinning methods are considered, implemented and finally an evaluation is carried out obtaining many conclusions related to eficience, speed and requirements, as well as ideas for future implementations.
In the first part of the document, some definitions are presented related to the used methods, then the obtenied results are shown over the same set of images applying the proposed theories and finally, some ideas about how to optimized the chosen algorithms are exposed.

Resumen: el proceso de reconocimiento de la escritura manuscrita forma parte de las iniciativas que propenden a la preservación de patrimonio cultural resguardado en Bibliotecas y archivos donde existe una gran riqueza de documentos y hasta fichas manuscritas que acompañan libros incunables. Este trabajo es el punto de partida de un proyecto de investigación y desarrollo orientado a la digitalización y reconocimiento de material manuscrito y la ponencia que aquí se presenta discute diferentes algoritmos utilizados en una primera etapa dedicada a "limpiar" la imagen de ruido para mejorarla antes de comenzar el reconocimiento de caracteres. Dado que PrEBi-SeDiCI forman parte integrante de redes de bibliotecas que intercambian documentos digitalizados vía scanning, el presente desarrollo ha tenido una utilización adicional relacionada al mejoramiento de las imágenes de documentos de intercambio que presentaban problemas comunes en la digitalización: bordes, impurezas, descentrado, etc.., si bien no es esta la finalidad de esta investigación no por ello resulta una utilidad menor en el marco de intercambios de consorcios de bibliotecas.
Para que el proceso de digitalización y reconocimiento de textos manuscritos sea eficiente debe estar precedido de una etapa de "preprocesamiento" de la imagen a tratar que incluye umbralización, limpieza de ruido, adelgazamiento, enderezamiento de la línea base y segmentación

[1] Investigador Comisión de Investigaciones Científicas de la Provincia de Buenos Aires – CIC y Directora del Proyecto de Enlace de Bibliotecas (PrEBi) y del Servicio de Difusión de la Creación Intelectual (SeDiCI) de la Universidad Nacional de La Plata, Argentina. Dirección de consulta: marisadg@ing.unlp.edu.ar

[2] Becario del Proyecto de Enlace de Bibliotecas (PrEBi) y del Servicio de Difusión de la Creación Intelectual (SeDiCI) de la Universidad Nacional de La Plata, Argentina.
[3] Becario del Proyecto de Enlace de Bibliotecas (PrEBi) y del Servicio de Difusión de la Creación Intelectual (SeDiCI) de la Universidad Nacional de La Plata, Argentina.

de la imagen entre otros. Cada uno de estos pasos permitirá reducir la variabilidad nociva al momento de reconocer los textos manuscritos (ruido, niveles aleatorios de grises, inclinación de caracteres, zonas con más y menos tinta), aumentando así la probabilidad de reconocer adecuadamente los textos. En este trabajo se consideran dos métodos de adelgazamiento de imágenes, se realiza la implementación y finalmente se lleva adelante una evaluación obteniendo conclusiones relativas a la eficiencia, velocidad y requerimientos, así como también ideas para futuras implementaciones.

En la primera parte del documento, se presentan algunas definiciones relacionadas con los métodos utilizados, luego se muestran los resultados obtenidos sobre un mismo conjunto de imágenes aplicando las teorías propuestas y finalmente, se exponen algunas ideas para optimizar los algoritmos elegidos.

Palabras Claves: conservación patrimonial, digitalización, adelgazamiento, componentes conexas.

Introducción: En los últimos años la digitalización de piezas documentales custodiadas en bibliotecas y archivos de todo el mundo ha tomado una enorme significación por sus infinitas posibilidades, tanto para los especialistas como para un público más amplio. Este nuevo espacio social, posibilitado por la tecnología digital permite descubrir un conjunto de objetos que constituían una riqueza cultural poco conocida. Sin embargo, las ventajas actuales de esta tecnología también han representado un duro aprendizaje que ha dejado tras de sí errores cruciales que hemos pagado con la destrucción o el deterioro considerable de piezas originales. Por esta razón de orden práctico, el problema de la digitalización y su propia naturaleza tecnológica, ha abierto un espacio de reflexión internacional que se ha caracterizado primordialmente por el constante flujo de información especializada y por presentar un permanente discurso de beneficio social.

Al ser digitalizados, los documentos son mejor preservados pero la búsqueda y el acceso a la información allí contenida es un proceso lento, secuencial y por consiguiente, altamente ineficiente. La catalogación de dichos documentos puede solucionar el problema, pero requiere una gran cantidad de recursos humanos y temporales para llevar a cabo ese proceso; la indexación automática de los documentos digitalizados, utilizando tanto sistemas de reconocimiento de patrones como sistemas gestores de bases de datos, permite acelerar el proceso de catalogación así como también el de búsqueda y acceso a los datos, convirtiéndola en una solución viable en cuanto a complejidad, tiempos y costo se refiere.

Para reconocer la escritura manuscrita en un documento digitalizado deben realizarse un conjunto de operaciones que permitan caracterizar individualmente cada uno de los objetos extrayendo la mayor cantidad de características posibles de cada elemento. Para que este proceso de extracción de características sea lo mas eficiente posible, debe realizarse un procesamiento previo de la imagen, eliminando todo aquello que genere ambigüedades o confusiones entre objetos similares. Una de las tareas más importantes a realizar es la obtención del esqueleto de los objetos, lo cual se logra adelgazando dichos objetos manteniendo su tamaño y su forma.

Definiciones previas.

Una imagen puede verse como un conjunto de puntos que poseen un determinado valor que indica el color del mismo.[4] En las imágenes en tonos de grises esos valores oscilarán entre 0 (punto negro – sin brillo) hasta 255 (punto blanco).

Las imágenes con las que trataremos ya han sido binarizadas (umbralizadas), aplicándoles tratamiento previo en el cual, en base a un valor umbral, se les ha asignado 0 a todos los píxeles que superen ese valor y 1 a todos aquellos que están por debajo del mismo. La nueva imagen constituida por 1's y 0's permite trabajar en el reconocimiento de la escritura eliminando el problema de los colores o tonos de grises. La elección del umbral puede ser manual o automática, existiendo muchos

[4] Para imágenes a color, cada punto almacena 3 valores: nivel de rojo, de azul y de verde (RGB). El lector interesado puede leer las referencias [10][11]

métodos para elegir el mejor umbral. Si bien una exposición detallada excede los propósitos de este atículo, el lector interesado puede consultar la referencia [11].
El conjunto de píxeles (también llamados puntos p = (x,y) con x , y ∈ N) que forman una imagen puede verse de dos maneras[5]:

a)Como una matriz, donde I [p] = I [(x,y)] = k; con k valor del pixel x,y (k ∈ [0,255] en imágenes grises; k ∈ [0,1] en imágenes binarizadas)
b)Como una función Pixel: (Imagen , punto) -> valor
donde Pixel(I , p) = Pixel(I , (x,y)) = k.

Para **adelgazar una imagen manteniendo la topología** (de modo de no alterar las formas) se deberán eliminar puntos que pertenezcan al borde, o sea, que cumplan las siguientes propiedades:

1)Punto negro (no tiene sentido eliminar puntos blancos!)
2)Punto simple
3)Exclusión de punto final

Donde:

- Un píxel P se considera *punto simple* si el conjunto de los vecinos negros de P tiene **una sola componente conexa** adyacente a P.
- Un píxel P se considera *punto final* si posee un solo vecino negro (punto extremo de la imagen).

Para eliminar los puntos del borde, se deberán realizar sucesivas pasadas (manteniendo de éste modo la topología). En cada pasada, deberán marcarse todos los puntos “eliminables” y luego proceder con la eliminación. Las pasadas continuarán mientras de esa operación resulte la eliminación de al menos un píxel.

Concepto de Vecindad.

Los vecinos de un píxel están condicionados al tipo de mallado utilizado para representar la imagen.

Se define q-vecindad (también q-adyacencia) como el conjunto de píxeles vecinos de p. El valor de q dependerá del tipo de mallado. Si se utiliza un mallado hexagonal, cada píxel de la imagen (menos los bordes) posee 6 vecinos, en cuyo caso se habla de una 6-vecindad. Un mallado cuadrangular presenta dos posibles opciones: una 4-vecindad (considerando como vecinos a los píxeles que se encuentran a los lados, arriba y abajo del pixel) o una 8-vecindad (idem 4-vecindad más los 4 píxeles de las diagonales).

Se define también (p,q)-adyacencia como:

p-adyacencia para píxeles negros

q-adyacencia para píxeles blancos[6]

Cálculo de componentes conexas en una imagen.

[5]*En el presente trabajo se prefiere la notación funcional, utilizando expresiones matemáticas y en algunas oportunidades incluyendo matrices en los casos de conceptos tales como vecindad y adyacencia.*

[6] La *q-distancia* entre dos píxeles se conoce como la longitud del camino más corto que los une.

Una componente conexa (CC) de una imagen es un conjunto de píxeles que cumplen que, para todo par existe un camino digital que lo une. Un **camino digital** de un pixel p a otro pixel q es un conjunto de píxeles $P_{pq} = \{ p_i \ i=0..n\}$ tal que:

1)$\forall\ p_i \in Ppq$, $color(p_i) = color(p) = color(q)$
2)$p0 = p$; $pn = q$
3)Para todo i=1,...,n-1, pi tiene exactamente dos vecinos en P_{pq} que son pi-1 , pi+1
4)*p_0, p_n tienen exactamente un vecino que son p1 y pn-1, respectivamente.*

Finalmente, una componente conexa está **acotada** si no posee ningún pixel del borde.

Algoritmo para calcular las componentes conexas (se usa 4-adyacencia).

Consideraciones:
La imagen que se analiza ha sido binarizada. Los píxeles valen 1 (negro) ó 0 (blanco).
Se utiliza una matriz, del mismo tamaño que la imagen que almacenará etiquetas a utilizar para caracterizar los píxeles de la imagen original de acuerdo a su clase de equivalencia.

Se recorre la imagen de izquierda a derecha y de arriba a abajo. Para cada pixel p=(x,y) se examinan los vecinos p1=(x-1,y) y p2=(x,y-1).

- Si ambos valen 0 (son blancos) se crea una nueva etiqueta y se le asigna a P en la matriz de etiquetas

- Si solo uno es 0, entonces se le asigna a P la misma etiqueta del otro (el que no es cero)

- Si ninguno es cero, se le asigna a P la etiqueta de cualquiera de los dos. Pueden presentarse dos escenarios:
    - La etiqueta de p1 y la de p2 son iguales, con lo cual la etiqueta de p será igual a ambas
    - Las etiquetas de p1 y p2 son distintas, en este caso a p se le da la etiqueta de cualquiera de las dos. Supongamos que se le da la etiqueta de p1. Entonces deberá registrarse que aunque p2 y p poseen diferentes etiquetas, pertenecen a la misma componente.

Si se consideran las etiquetas como clases de equivalencia a las que pertenecen los píxeles, si p1 pertenece a la clase 1 y p2 a la clase 2, y durante el procesamiento de la imagen se ha registrado que p1 y p2 pertenecen a la misma componente, entonces la clase de equivalencia de p1 es la misma que la de p2. Por lo tanto, Clase 1 = Clase 2. Para mantener este registro se utiliza un vector en memoria que posee tantos lugares como etiquetas han sido utilizadas. Las etiquetas son número enteros lo que permite utilizarlas para acceder directamente al vector como índices).
Cada posición del vector representará la etiqueta y su contenido la clase de equivalencia a la que pertenece esa etiqueta. Por ejemplo, si se tiene el vector:

V = {1,2,3,2,3,1,4}

La primera posición del vector es la posición 0, la última es la n-1 (en nuestro caso, 6) y la semántica del vector V es:
"la etiqueta 'x' pertenece la clase de equivalencia V[x]"
Luego:

- Las etiquetas 0 y 5 pertenecen a la clase de equivalencia 1
- Las etiquetas 1 y 3 pertenecen a la clase de equivalencia 2
- Las etiquetas 2 y 4 pertenecen a la clase de equivalencia 3
- La etiqueta 6 pertenece a la clase de equivalencia 4

Lo que significa: sea Eq :: Etiqueta -> Entero una función que toma una etiqueta y retorna la clase de equivalencia a la que pertenece esa etiqueta;

Eq(x) = V[x];

Con este procedimiento se ha obtenido una matriz que contiene las etiquetas de todos los píxeles de acuerdo a sus clases de equivalencia. Varias etiquetas que pertenecen a la misma clase obligan a la normalización de la matriz, recorriendo la misma y asignándole a cada elemento su clase de equivalencia.
Este algoritmo también almacenará todos los elementos que pertenecen a cada clase, por lo cual una opción sería mantener esta matriz en memoria. Esto generaría un gran desperdicio de la memoria, ya que también se reserva espacio para los espacios en blanco (aquellos píxeles que no poseen valor y que no pertenecen a ninguna clase de equivalencia). Como contrapartida, esta solución permite acceder a la clase de equivalencia de un pixel determinado de manera directa.

Sea p = (x,y) pixel ∈ I imagen.

Eq(p) = I(x,y);

(cabe recordar que las imágenes pueden ser vistas como funciones que toman dos valores enteros x e y, y retornan el valor del pixel en la posición (x,y)).
Otra solución planteada consiste en almacenar en el vector de las clases de equivalencias existentes otro vector, donde por cada posición se almacenan todos los puntos que pertenecen a esa clase de equivalencia. La definición de tipo sería

```
Type
   Punto = record
          X,Y : integer // píxeles
           End;

  VectorDePuntos = array of punto;
   Clase = record
           puntos : VectorDepuntos
           numeroDeClase : integer
          end;
  Clases = array of Clase
```

La principal ventaja de ésta implementación radica en el ahorro de memoria, ya que solo se almacenarán aquellos puntos que pertenezcan a alguna clase de equivalencia. Su principal desventaja es que encontrar la clase de equivalencia de un determinado punto implicará recorrer el vector de clases de equivalencia, y para cada clase, recorrer el vector de puntos que pertenecen a la misma. En el mejor caso, esto significará un solo acceso (primera clase, primer punto). En el peor caso:

Si todos los puntos de una imagen determinada I pertenecen a alguna clase de equivalencia y la imagen I posee x píxeles de alto por y de ancho en un total de z = x*y. En el peor caso se considerará que el punto buscado se encuentra en la última posición del vector y en la última clase de equivalencia, y el acceso a este punto demandara z accesos a memoria.

El “peor” caso no es 100% realista, ya que es imposible que todos los puntos de una imagen pertenezcan a alguna clase de equivalencia pues significaría que la misma posee, por ejemplo, todos los píxeles negros. A partir de la experiencia adquirida, es posible estimar (para el tipo de imágenes trabajadas) que menos de la mitad de los píxeles son negros y en algunos casos la cifra sólo llega al 20%.

Algoritmo de adelgazamiento de Zhang-Suen

Este método es rápido y sencillo de implementar, consta de dos subiteraciones en cada una de las cuales se eliminan aquellos píxeles que cumplan con todas las reglas definidas para la iteración.

Recordando que:

- Un píxel es un **punto final** si tiene un único vecino de color negro, siendo todos los demás blancos.
- La **conectividad** de un píxel se define como el número de objetos que podría conectar en la imagen original y se calcula girando alrededor de un píxel en el sentido de las agujas del reloj y contando cuántos cambios de color se producen. El número de cambios será la conectividad, es decir, el número de regiones que une.

Como primer paso del algoritmo, se aplica un suavizado de la imagen, se borran todos los píxeles que tengan dos o menos vecinos negros y conectividad menor a dos.[7]

Luego se realizan las dos iteraciones.

Para eliminar un píxel en la primera iteración, el mismo debe cumplir con las siguientes propiedades:

- Tener conectividad 1
- Cantidad de vecinos negros entre 2 y 6 (incluídos)
- Al menos uno de los siguientes píxeles blanco: [x-1,y], [x,y+1], [x,y-1]
- Al menos uno de los siguientes píxeles blanco: [x-1,y], [x+1,y], [x,y-1]

En la segunda iteración se eliminarán los píxeles que cumplan las siguientes reglas:

- Tener conectividad 1
- Cantidad de vecinos negros entre 2 y 6 (incluídos)
- Al menos uno de los siguientes píxeles blanco: [x-1,y], [x,y+1], [x+1,y]
- Al menos uno de los siguientes píxeles blanco: [x,y+1], [x+1,y], [x,y-1]

Como expresamos anteriormente, las iteraciones continuarán mientras se eliminen puntos. Si no se realiza el pre-procesamiento de la imagen, los resultados no son tan buenos, los resultados se muestran en las Figura 1 y 2.

[7] Notar que esto requiere un procesamiento extra, reduciendo la eficiencia del algoritmo.

a) b) c)

Figura 1: a) Imagen original b) Esqueleto de la imagen obtenido con el algoritmo basado en Componentes Conexas c) Esqueleto obtenido con el método de Zhang-Suen.

a) b) c)

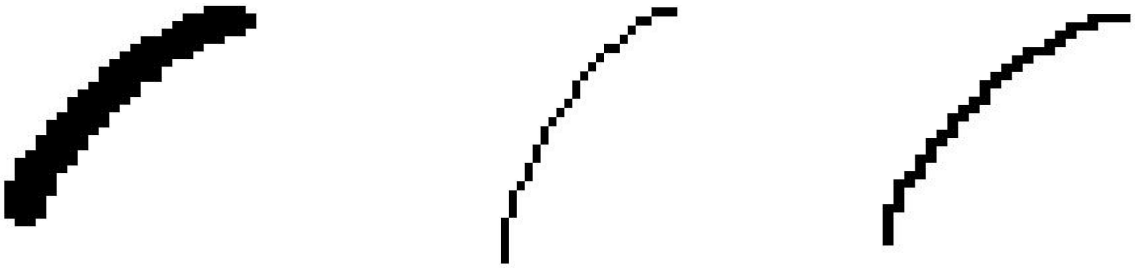

Figura 2: Similar a la Figura 1). Aquí se puede observar en detalle como se generan las líneas en ambos métodos, se aprecia un grosor superior en el caso del método de Zhang-Suen.

Problema en los bordes con ruido.

De acuerdo a los experimentos realizados, los resultados no han sido tan buenos con ciertas imágenes, por ejemplo en aquellos casos donde los bordes están difuminados o poseen ruido aleatorio, en cuyo caso, el proceso de adelgazamiento produce líneas cortas aleatorias, generando una deformación de la imagen, o lo que es lo mismo, una pérdida de su forma. Los problemas expuestos en este tipo de imágenes conllevan dificultades a la hora del reconocimiento de caracteres.

Las imágenes que presentan esta característica poseen un borde “sucio” con puntos distribuidos aleatoriamente y al adelgazarlas, los puntos al azar no son eliminados ya que cumplen con las características de punto del borde. Ambos algoritmos reaccionan del mismo modo ante este fenómeno generando líneas a lo largo de todo el borde de la imagen, similares a ramificaciones del esqueleto de la misma, en un intento por mantener unidos los puntos del borde con dicho esqueleto (ver Figura 3).

Una solución simple.

Este problema puede ser solucionado aplicando un filtro medio a la imagen antes de adelgazarla produciendo bordes más lisos y sin las imperfecciones anteriores. En los resultados obtenidos (ver Figura 4) se han conseguido esqueletos de imágenes claramente mejores, especialmente con el algoritmo de Zhang Suen, ya que con el método basado en CC no se obtienen siempre las mismas mejoras. Como desventaja, éste método agrega un costo extra de procesamiento.

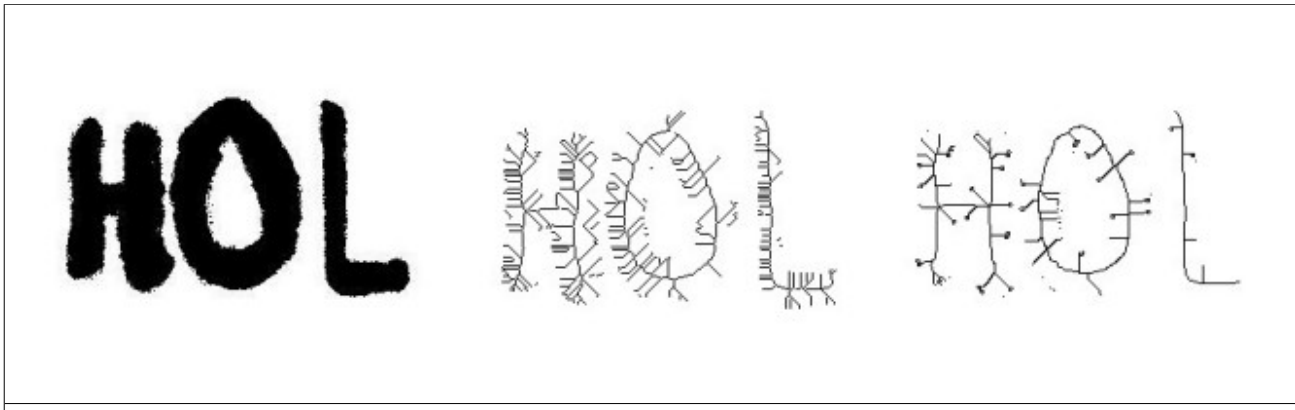

Figura 3 Esqueletos de la primer imagen obtenidos con CC en el primer caso y con Zhang-Suen en el segundo caso.

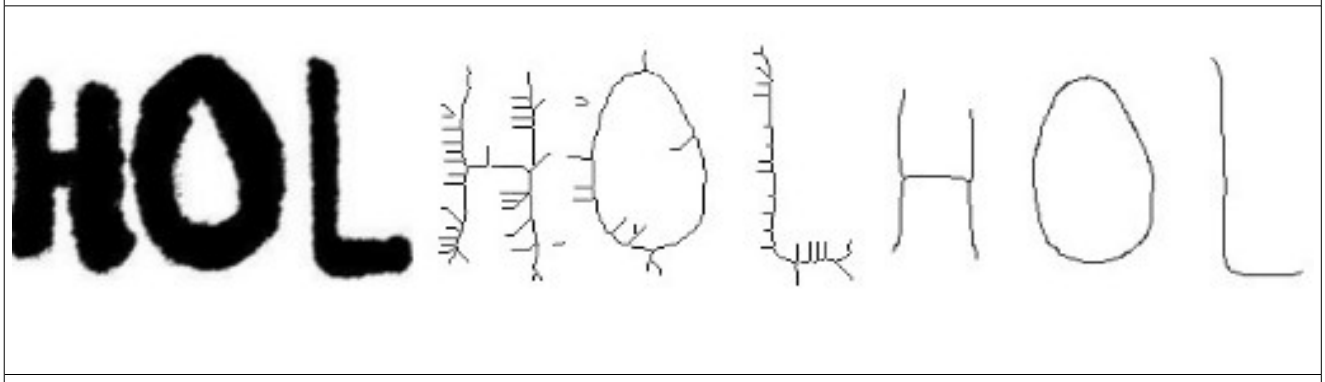

Figura 4 Imagen a la que se aplicó el filtro medio; Esqueleto de la misma imagen con CC; esqueleto de dicha imagen con Zhang-Suen

Optimización del método basado en Componentes Conexas.

El problema del algoritmo de adelgazamiento basado en el cálculo de componentes conexas es su elevada complejidad e ineficiencia., puede ser mejorado utilizando otro método para reconocer los puntos simples (principal uso del algoritmo) sin la necesidad de obtener todas las componentes conexas. Esta técnica tiene la misma base que el método basado en componentes conexas: los puntos que se podrán eliminar son aquellos que son simples y no son punto final, pero requiere un tiempo considerablemente menor para determinar si un punto es simple.

La idea de esta optimización consiste en verificar cada punto en un determinado lado; esto quiere decir que un punto puede verse como punto simple en una dirección si sus vecinos cumplen con ciertas condiciones **en esa misma dirección**.

El algoritmo realiza varias pasadas, verificando en cada una un lado distinto: lado norte, sur, este y oeste. En cada una de las pasadas se realiza un recorrido por la imagen buscando aquellos puntos negros que son simples de acuerdo al sentido en que se esté recorriendo y que no sea punto final.

Sea d la dirección en que se está analizando el punto P. Llamaremos P_d al conjunto de las condiciones que indicarán si el punto P es punto simple en el sentido d. Desde luego, para cada d posible (norte, sur, este y oeste), el conjunto P_d será distinto. A los elementos del conjunto P_d los llamaremos funciones c_i

c_i : (Imagen, Punto) -> [F,T] (F = FALSE, T = TRUE)

La estructura del conjunto P_d es similar para todos los casos: existen un conjunto de condiciones que **NO** deben cumplirse otras que deberán cumplirse necesariamente. Una condición común es que el punto P tenga el valor negro, la cual será ignorada aquí como miembro del conjunto (ya que siempre se considera que se están analizando puntos negros).

Sea I(x,y) : Imagen -> [0,1] *función que retorna el valor del punto (x,y). (1= blanco, 0 = negro)*
Sea Sn:: Punto -> [F,T] Sn es la función que indica si un punto es simple norte. (F = FALSE, T = TRUE)
d = Norte

$$Sn(p) \begin{cases} T \text{ si } I(x-1, y)= 1 \wedge \neg\exists\, c_{i\in} P_d \\ c_i = T \text{ con } i=1, 2,3,4,5 \end{cases}$$

- c1= I(x,y-1) =0 ^ I(x+1,y) =1 ^ I(x,y+1) =0
- c2= I(x,y-1) =1 ^ I(x-1,y-1) =0 ^ I(x-1,y) =1
- c3= I(x-1,y) =1 ^ I(x-1,y+1)=0 ^ I(x,y+1) =1
- c4= I(x,y+1)=1^ I(x+1,y+1)=0 ^ I(x+1,y) =1
- c5= I(x+1,y)=1^ I(x+1,y-1) =0 ^ I(x,y-1) =1

En la Figura 5 pueden verse gráficamente las 5 condiciones.

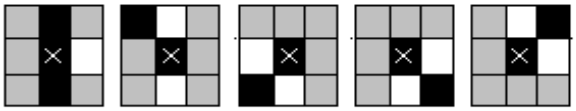

Figura 5: Casos que permitirán identificar un punto simple norte.

Los puntos negros y blancos son los puntos que deberán poseer ese valor. Los puntos grises son puntos en los que su valor no es de interés. El punto marcado con una (x) es el punto que está siendo analizado.
Análogamente, se define Ss(p) (Punto Simple Sur), Se(p) (Punto Simple Este) y So(p) (Punto Simple Oeste) como:

d = Sur

$$Ss(p) \begin{cases} T \text{ si } I(x+1, y)= 1 \wedge \neg\exists\, c_{i\in} P_d \\ c_i = T \text{ con } i=1, 2,3,4,5 \end{cases}$$

- c1 = I(x-1,y)=1 ^ I(x,y+1)=0
- c2,c3,c4 y c5 se mantienen iguales

d = Este

$$Se(p) \begin{cases} T \text{ si } I(x, y+1)= 1 \wedge \neg\exists\, c_{i\in} P_d \\ c_i = T \text{ con } i=1, 2,3,4,5 \end{cases}$$

- c1 = I(x-1,y-1)=1 ^ I(x-1,y)=0 ^ I(x+1,y)=0
- c2,c3,c4 y c5 se mantienen iguales

d = Oeste

$$So(p) \begin{cases} T \text{ si } I(x, y-1)= 1 \wedge \neg\exists\, c_{i\in} P_d \\ c_i = T \text{ con } i=1, 2,3,4,5 \end{cases}$$

- c1 = I(x,y+1)=1 ^ I(x-1,y)=0 ^ I(x+1,y)=0
- c2,c3,c4 y c5 se mantienen iguales

Como se habrá observado, las condiciones c2,c3,c4 y c5 son siempre las mismas y han sido incluidas en el conjunto P_d por una cuestión de implementación del algoritmo, en donde se realiza un *or lógico* entre todas las condiciones del conjunto (luego ese resultado es negado para simbolizar el ¬∃ de la definición de cada función)

Una vez encontrados los puntos que cumplen las condiciones de punto simple (en la dirección que corresponda) y no final, se eliminan cambiando su valor a blanco. Esto se repetirá, al igual que en el algoritmo anterior, mientras se encuentren puntos que cumplan alguna condición

Bordes distintos.

En la Figura 6 se muestran imágenes, dos de las cuales han sido adelgazadas con cada uno de los algoritmos descriptos; el adelgazamiento en ambos (b y c) casos ha sido correcto, manteniendo la topología y sin efectos indeseados, pero si se observa con un nivel superior de detalle, podrá verse que la línea que genera el algoritmo de CC es más delgada que la línea que genera Zhang-Suen.
En la Figura 6 se muestran tres imágenes. La primera es una línea adelgazada usando CC, y la tercera es la misma línea, pero se utilizó Zhang-Suen. En la imagen central se pueden apreciar en color gris cuales son los nuevos puntos que aparecen al utilizar el último algoritmo.

Figura 6: Las líneas 1 y 3 han sido adelgazadas con los métodos de CC y Zhang-Suen respectivamente. La línea central muestra la "diferencia" entre ambas.

Esto se debe a que, al usar 4-adyacencia en CC, para mantener una línea unida (dentro de la misma CC) solo deberán tocarse sus vértices. Si bien en este caso el problema no pasa a mayores, en muchas situaciones es deseable que las líneas estén más definidas. La solución a esta situación particular es bastante simple: en vez de tomar 4-adyacencia, utilizar 8-adyacencia al hallar las CC; o sea, en vez de considerar solo dos puntos se deberán considerar 4. En la Figura 7 se muestran en gris, en primer lugar, los puntos que se consideran con 4-adyacencia mientras que a continuación, se presentan con 8-adyacencia.

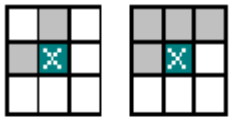

Figura 7: Píxeles a analizar si se toma 4-adyacencia y 8-adyacencia.

Al utilizar 8-adyacencia, se obtienen resultados similares en ambos algoritmos.

Conclusiones:
Si bien ambos algoritmos producen resultados adecuados, el algoritmo de Zhang-Suen demostró ser más sencillo de implementar y, en materia de eficiencia empírica, se observó que generalmente éste algoritmo demanda menores tiempos de ejecución que el algoritmo basado en CC, incluso en su versión mejorada. Los datos pueden apreciarse en las Figuras 8 y 9, donde se observa gráficamente los tiempos que tomo la ejecución de cada algoritmo en su versión simple así como también luego de aplicar filtros.

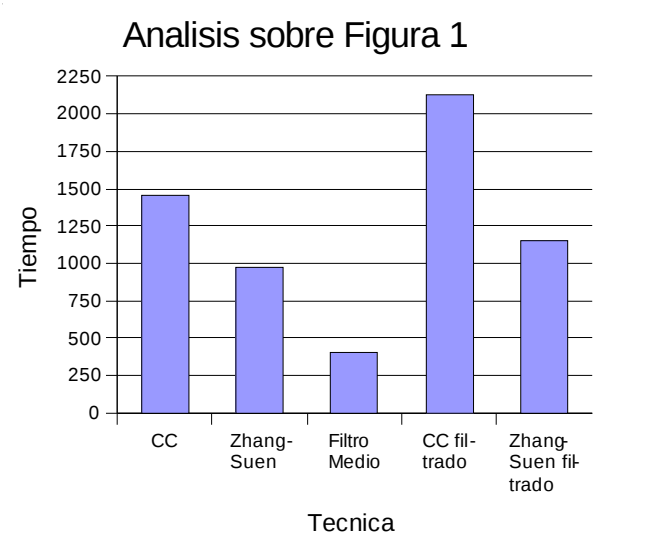

Figura 8 Análisis de los tiempos para las distintas técnicas desarrolladas sobra la imagen de la Figura 1.

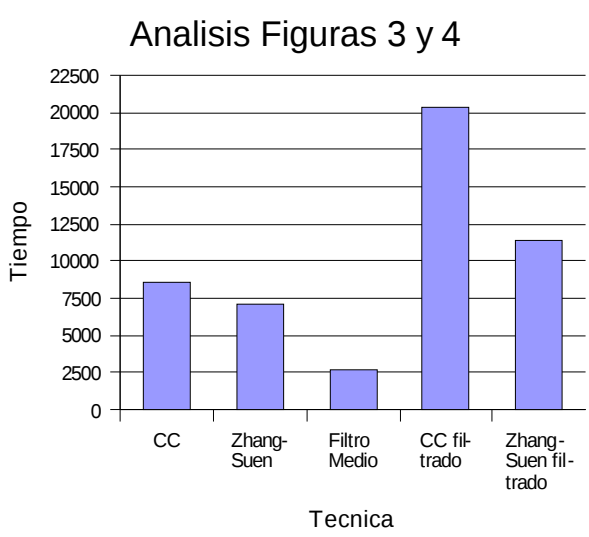

Figura 9: Diagrama de tiempos aplicadas a las Figuras 3 y 4 con los distintas técnicas.

No quiere decir que sea siempre mejor utilizar el primer algoritmo. Uno de los problemas hallados con el algoritmo de Zhang-Suen es que si existen líneas gruesas que se intersecan, al adelgazarlas se genera un efecto en el cual la intersección produce un segmento en vez de un simple punto. A esto se lo conoce como efecto *necking*; y para solucionarlo se debe hacer un preprocesado de la imagen, poniéndole más énfasis sobre sus ángulos cerrados. Pero esto también implica más procesamiento, lo que se traduce en tiempo extra de ejecución.
Por otro lado las componentes conexas de una imagen son utilizadas también para otros fines, por ejemplo, conocer la cantidad de zonas cerradas en una imagen (agujeros), invirtiendo la imagen (cambiando blancos por negros y negros por blancos). Luego, todos los puntos que pertenezcan a la misma CC, pertenecerán a la misma zona encerrada. Y la cantidad total de zonas cerradas será la cantidad total de componentes conexas menos 1. El método también es utilizado como base para segmentar la imagen en palabras, tomando en principio elementos de la misma componente conexa como elementos simples.
Si se toma una visión global del sistema, utilizar componentes conexas puede beneficiar la **eficiencia**, ya que se calculan solo una vez y luego se utilizan tantas veces como se requiere.

Referencias:

[1] Elisabetta Bruzzone, Meri Cristina Coffetti; Elsag spa – R&P Departament. **An algorithm for**

Extracting Cursive Text Lines, Genova – Italy
[2] Radmilo M. Bozinovic, Sargur N. Srihari; IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. VOL 11 NO. 1 **Off-Line Cursive Scrtipt Word Recognition**
[3] K. Badie and M. Shimura. **Machine recognition of roman cursive script** in Proc. 6th. Int. Conf. Patternt Recognition, Munich, West Germany, Oct 1982.
[4] R. Manmatha, Chengfeng Han, E. M. Riseman and W. B. Croft; **Indexing Handwriting Using Word Matching** Center for Intelligent Information Retrieval, Computer Science Department, Univeristy of Massachusetts, Amherst.
[5] Toni M. Rath and R. Manmatha; **Features for Word Spotting in Historical Manuscipts**. Multi-Media Indexing and Retrieval Group. Center for Information Retrieval. University of Massachusetts, Amherst.
[6] L. Fletcher and R. Kasturi. **A robust algorithm for text string separation from mixed text/graphics images.** IEEE Transactions on Pattern Analysis and Machine Intelligence, 10:910-918. 1988
[7] F. Wahl, K. Wong and R. Casey. **Block segmentation and text extraction in mixed text/image documents**. Computer Vision Graphics and Image Processing, 20:375-390,1982
[8] D. Wang and S. N. Srihari. **Classification of newspaper image blocks using texture analysis**. Computer Vision Graphics and Image Processing, 47:329-352
[9] Michel Weinfeld, **Reconnaissance d'ecriture manuscrite: segmentation de mots.** Département d'Enseignement et de Recherche en Informatique. École Polytechnique, Paris, France
[10] William Prat, John Wiley & Sons, **Digital Image Processing**, 1991, Second Edition
[11] Gonzalez Rafael, Woods, Addison-Wesley **Digital Image Processing**, 1992, Second Edition
[12] T. M. Rath, R. Manmatha. **Word Spotting for Handwritten Historical Document Retrieval**