Capítulo 4. Normalización de Tamaño de Imágenes

Capítulo 4. Normalización de Tamaño de Imágenes

4.1 Procesamiento previo de las imágenes.

La segmentación de palabras manuscritas implica una serie de procesos previos necesarios para lograr cierta uniformidad entre las imágenes que son leídas por el segmentador. Esta uniformidad involucra una limpieza de las imágenes eliminando ruido (como renglones, sellos y firmas impresos en el documento original), y una estandarización de sus dimensiones, como lo son el ancho de las letras que contienen y su tamaño en el eje vertical [Grandidier & Sabourin, 1999].

Como se puede observar en la Figura 1.1 el preprocesamiento de las imágenes ocupa un papel muy importante en el proceso de reconocimiento porque la segmentación depende directamente de sus salidas. Algunos laboratorios de investigación han aplicado varios métodos semejantes a los implementados en el proyecto actual para eliminar la variabilidad entre las palabras [Kussul & Kasatkina, 1999].

Parte de los problemas del proyecto anterior a éste de reconocimiento de letra manuscrita de Porfirio Díaz se deben, en cierta medida, a una baja normalización de las imágenes que se leían. Cabe hacer notar que por normalización en este caso, nos referimos a que todas las imágenes de las letras o palabras a reconocerse sean del mismo tamaño, o casi del mismo tamaño. Esta falta de uniformidad dificultó la segmentación ya que el algoritmo utilizado se basaba en la densidad vertical de píxeles de la palabra. Esto

es, cada vez que se encontraba una densidad alta de píxeles sobre la imagen, se estimaba un caracter en esa posición. Mientras que si se encontraba una densidad baja se estimaba una ligadura (segmento que une a una letra con otra).

Los problemas comenzaron cuando se leían palabras con un ancho de letra muy grande y otras con un ancho muy pequeño. Entonces la idea de establecer un mismo umbral para determinar si una concentración de píxeles equivalía a una letra o a una ligadura resultaba una tarea difícil. De este modo si se establecía un umbral bajo se corría el riesgo de que las ligaduras de palabras estrechas fuesen ignoradas y tomadas como una sola letra de grandes proporciones. Si por el contrario se definía un umbral alto, las letras de palabras anchas eran consideradas como ligaduras debido a su baja densidad en columna.

Debido a estos problemas fue necesario agregar algunas mejoras al proceso de normalización del proyecto anterior para elevar los porcentajes de éxito finales.

4.2 Corrección de la inclinación de las palabras

La corrección de la inclinación de las palabras se llevó a cabo a dos niveles, con respecto a la línea de base de las palabras y con respecto al eje vertical. La figura 4.1 muestra claramente ambos enfoques. La palabras corregidas bajo este proceso son las mismas que se utilizaron en este proyecto [Linares & Spínola, 2000].

Figura 4.1 Diferentes tipos de inclinaciones
que presentan las palabras

La corrección de la inclinación con respecto a la línea de base se realizó a través del paquete de diseño Adobe Photoshop v 5.0. La línea de base es el renglón imaginario sobre el cual se encuentra escrita la palabra (esto es, a nivel horizontal). De no realizarse esta corrección la segmentación se vería afectada porque las ligaduras entre letras serían más difíciles de detectar ya que su longitud en el eje horizontal resulta muy pequeña.

La necesidad de hacer este tipo corrección surge a partir de que en algunos de los documentos originales no existían renglones de guía impresos, lo cual provocó que la línea de base presentara inclinaciones que resultan muy perjudiciales al momento de segmentar la palabra, Figura 4.2.

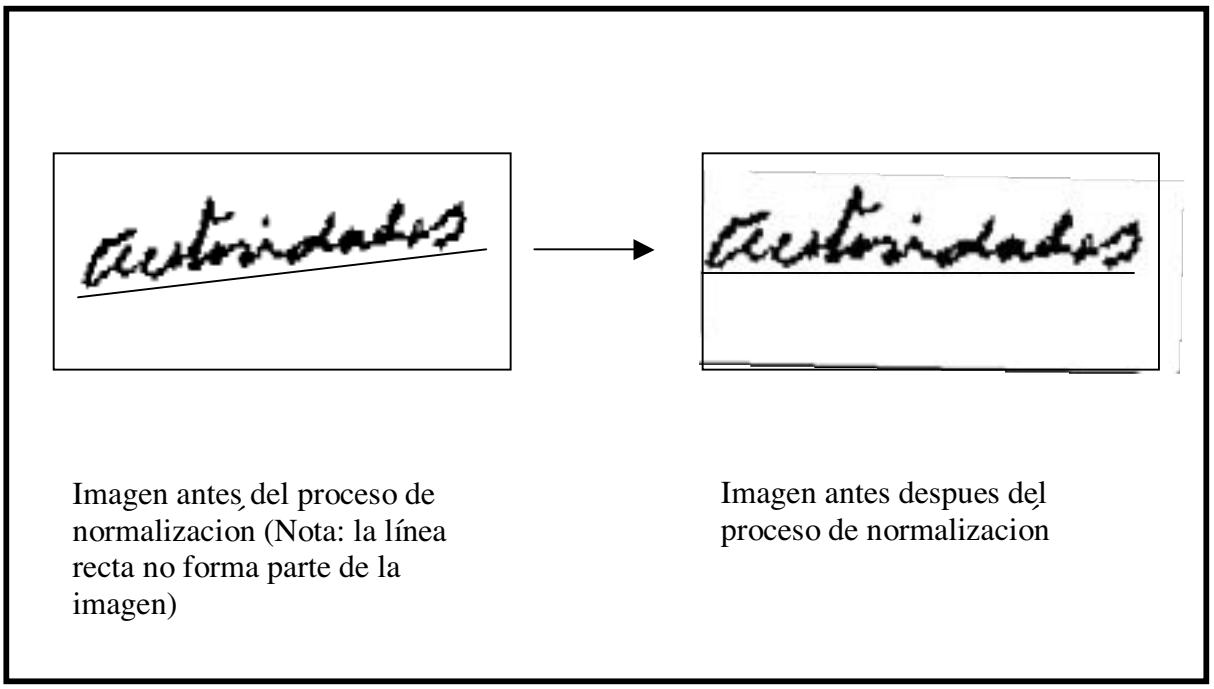


Imagen antes del proceso de normalizacion (Nota: la línea recta no forma parte de la imagen)

Imagen antes despues del proceso de normalizacion

Figura 4.2 Normalizacion de la inclinacion con respecto al eje horizontal.

El otro tipo de corrección que se aplicó fue con respecto al eje vertical. Aquí las variantes entre palabras residen en que muchas varían en su inclinación ya sea por cuestiones de características de la superficie sobre la cual se apoyaba el escritor o bien por comodidad del mismo. Como lo muestra la figura 4.3 algunas letras serían partidas de manera incompleta o con segmentos de otra.

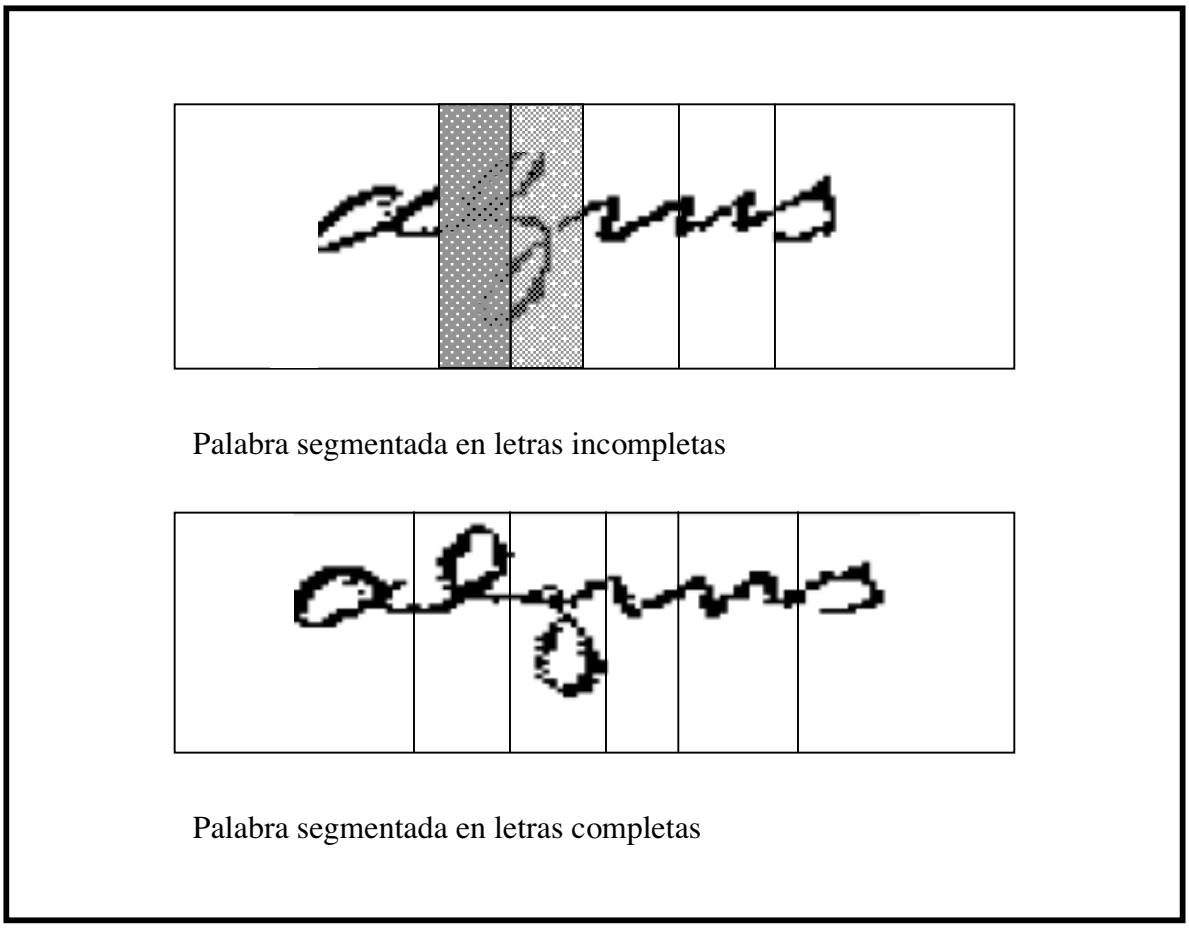


Figura 4.3 Normalizacion de la inclinacion de la palabra con respecto al eje vertical.

Es necesario precisar que todo este procedimiento se realizo de manera manual. De todos los artículos consultados a lo largo del proyecto, ninguno presenta un algoritmo que sea totalmente independiente a la percepcion de un humano. Sin embargo dıa a dıa se están realizando mejoras a este tipo de sistemas, lo cual hace factible que en un futuro la automatizacion de todo el proceso se realice sin necesidad de la intervencion humana [Kussul & Kasatkina, 1999]. En el proyecto anterior a las palabras a las que se les aplico este tipo de correccion se les guardo con el nombre de la palabra que representaban mas

el sufijo "_s" esto con la finalidad de establecer comparaciones finales entre el porcentaje de éxito que presenta el sistema con estas correcciones y sin ellas, dicha tabla se muestra en la figura 3.4.

4.3 Normalización vertical de las imágenes.

La normalización consiste en llevar a un tamaño estándar las dimensiones de una imagen, sin provocar en ella alguna distorsión de importancia [Aguilar & Toledo, 1995].

En el proyecto anterior no se estableció un procedimiento riguroso que asegurara que todas las letras tuviesen un mismo tamaño específico. Solamente se usó un algoritmo de escalamiento que aseguraba que todas las imágenes tuviesen el mismo tamaño, sin embargo existían una gran cantidad de píxeles blancos que rodeaban a la palabra. Entonces, el hecho de que se tuviesen imágenes del mismo tamaño no significaba que las letras que contenían estuviesen estandarizadas. En la figura 4.4 se puede observar este obstáculo.

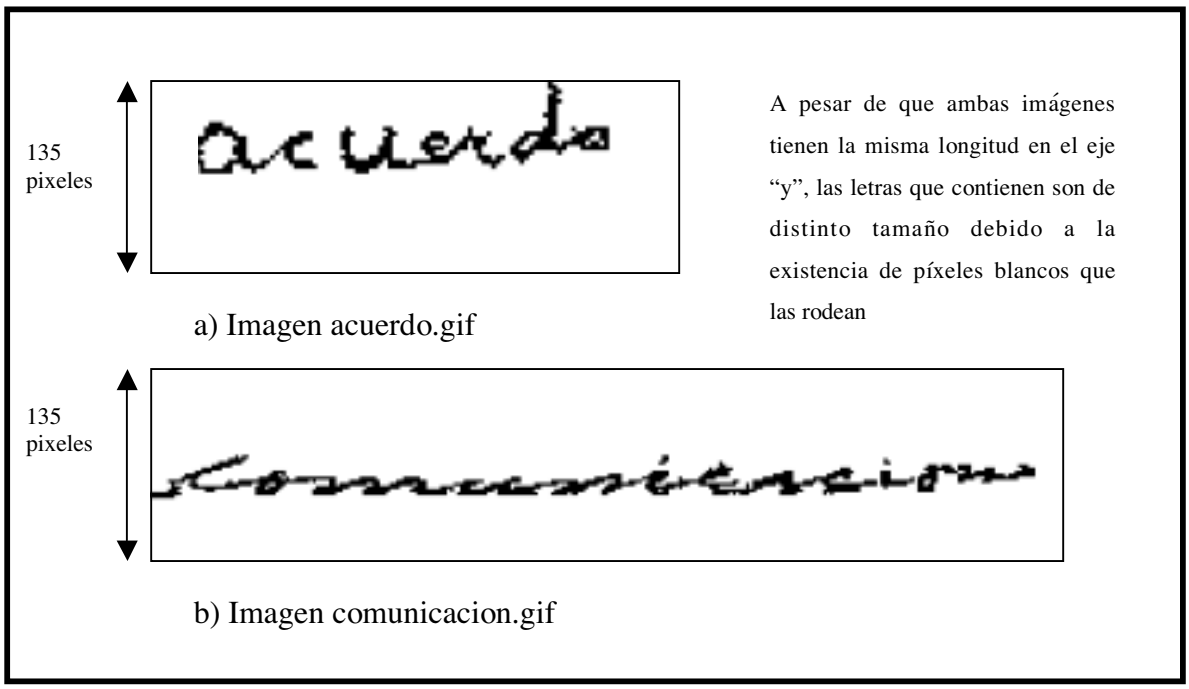


Figura 4.4 Ejemplos de tamanos de letras dentro
de imágenes de dimensiones idénticas

Como primera instancia se decidió en este proyecto eliminar los píxeles blancos que rodeaban a la palabra. Se aplicó un algoritmo de detección del píxel máximo y mínimo. El algoritmo consiste en encontrar el primer y último renglones que contienen un píxel negro. De esta manera se recorta la imagen eliminando píxeles innecesarios que resultan en una verdadera estandarización del tamaño de las letras al realizar un escalamiento.

Una vez que se han obtenido las imágenes recortadas el siguiente paso que se siguió fue el de establecer un escalamiento a un tamaño fijo. El primer algoritmo que se intentó aplicar para dicho escalamiento fue el propuesto por Gudsen [Aguilar & Toledo, 1995] el cual en un principio fue elegido ya que presenta un bajo índice de error y un

esfuerzo computacional aceptable. El algoritmo consiste en escalar cada columna de la imagen a un mismo tamaño. Se define WA como un vector de elementos (en este caso cada columna de la matriz de 1's y 0's) de entrada y también se define a WB como el vector de deseado (es decir, el tamaño deseado de la columna). Después se define un vector de elementos de tamaño WAxWB que es el tamaño que resulta de multiplicar el tamaño original por el tamaño de columna deseado. Entonces cada elemento de WA es almacenado WB veces en WAxWB. Después se forman grupos de WA posiciones y el contenido de tales posiciones se suma. Si esta suma rebasa un umbral preestablecido se considera como un píxel negro y si es inferior se considera como un píxel blanco de la nueva imagen. La Figura 4.5 esquematiza este procedimiento.

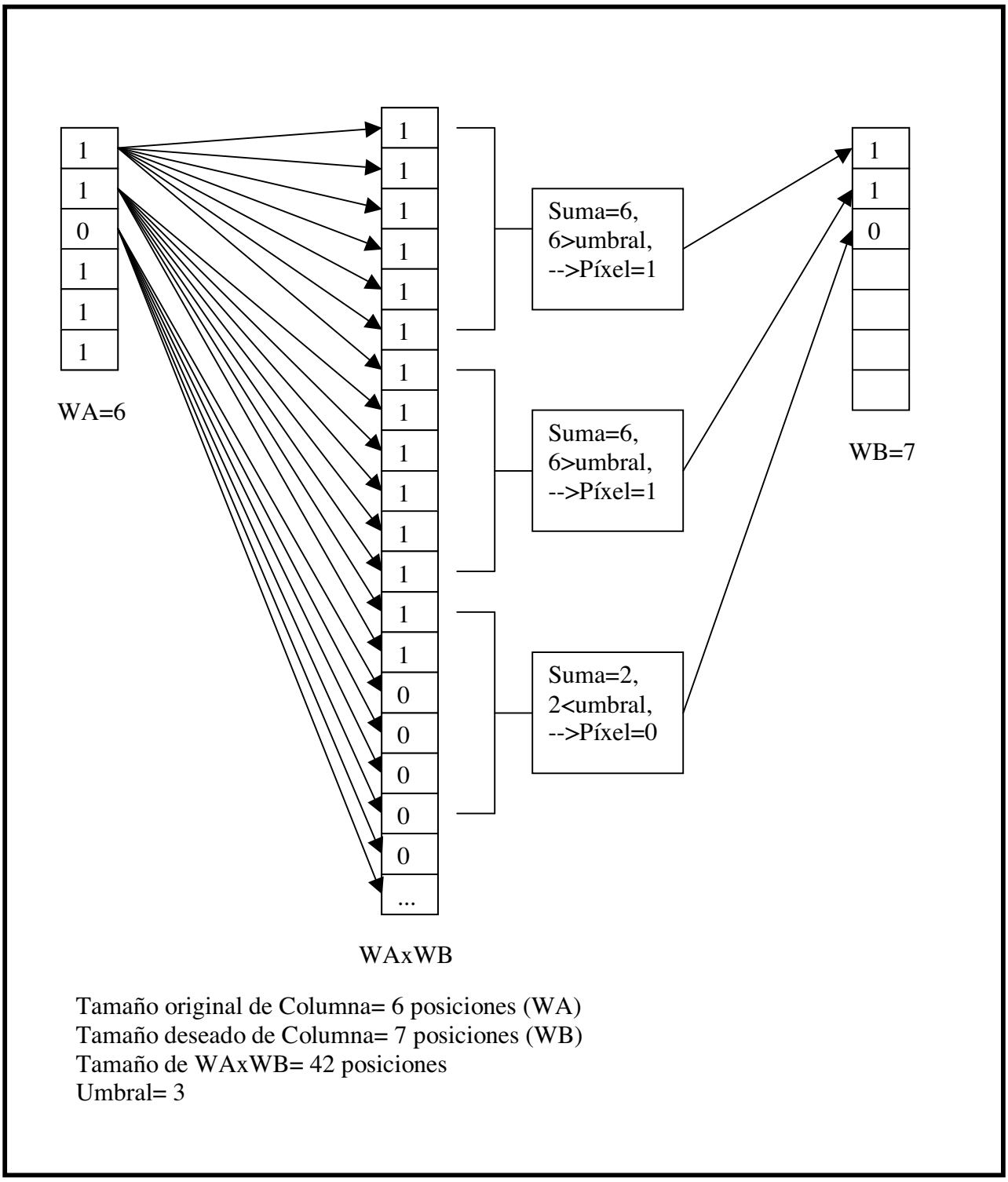


Figura 4.5 Diagrama del funcionamiento del Algoritmo propuesto por Gudsen [Aguilar & Toledo, 1995].

A pesar de las ventajas que ofrece este algoritmo en cuanto al índice de deformación de la imagen, presento una enorme desventaja en cuanto a su eficiencia ya que es mucho muy lento. Si se presenta una imagen (como es el caso de la base de datos que se esta utilizando) de tamaño original de 129 píxeles y se desea escalar a un tamaño de 135 píxeles (tamaño óptimo) se tendrían que almacenar en un vector temporal 135x129 píxeles, efectuar la suma de los mismos y almacenar su resultado en un vector de 135. Este mismo paso se tendría que efectuar por cada columna de la imagen. Esto ocasiona un almacenamiento en arreglos de 135x129x490 (número de columnas aproximadas por imagen)= 8,533,350 veces por imagen, lo cual representa un consumo excesivo.

Es por esta razón que se decidió usar los métodos de las clases proporcionadas por los paquetes de Java. La clase que se utilizó específicamente fue la clase *Image* contenida en el paquete java.Image.* . Esta clase ofrece un método de escalamiento *scale,* el cual realiza esta tarea de acuerdo con el algoritmo de escalamiento escogido por el usuario y que debe ser recibido como paramétro de entrada de este método. El algoritmo que se escogió fue Scale_Smooth, ya que permite el escalamiento aproximadamente en tiempo real sin presentar deformaciones notorias sobre la imagen. El resto de los algoritmos fue desechado ya que presentaban algunas deformaciones a la palabra escalada. Ver la Figura 4.6.

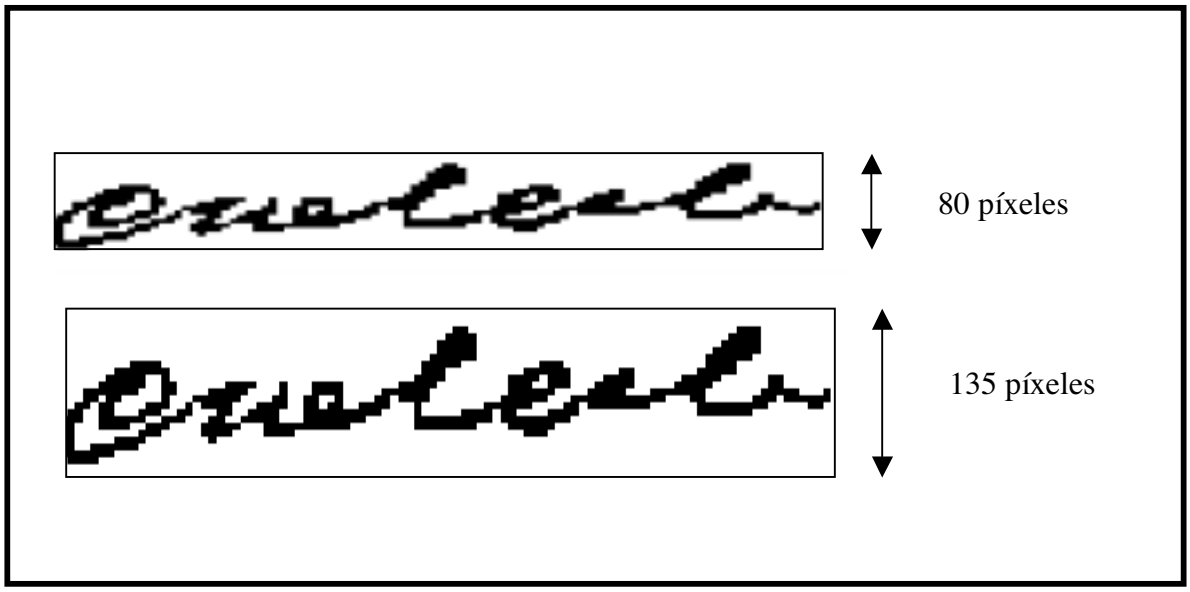


Figura 4.6 Escalamiento en el eje vertical usando algoritmos de java

4.4 Normalización horizontal de las imágenes.

Otro problema a resolver es que las imágenes manejan deferentes anchos de letras. Por ejemplo, si se tuviese la palabra “gobernador” escrita en dos imágenes diferentes es muy posible que en una, la letra presentara un ancho de 20 píxeles y en la otra imagen un ancho de 52. Esta variante se presenta ya que es común que un escritor cambie el tamaño de su letra debido a las características del documento sobre la cual escribe o bien para posicionarse de manera más cómoda. Esto trae repercusiones en la segmentación pues uno de los algoritmos que se utilizan dependen de la densidad de píxeles, como se explicó en la sección 4.1. Entonces el establecer un umbral fijo resulta muy complicado porque funciona sólo para ciertas palabras escritas de cierta manera y no funciona para palabras que sean mucho más espaciosas o más estrechas.

Por tanto se decidió un método que nos brindara una normalización con respecto al eje horizontal de la imagen cuyas variantes de tamaño son pequeñas. Se recurrió al método de histogramas presentado en [Kussul & Kasatkina, 1999] para obtener un número aproximado de puntos de segmentación de la palabra, como se muestra en la figura 4.7.

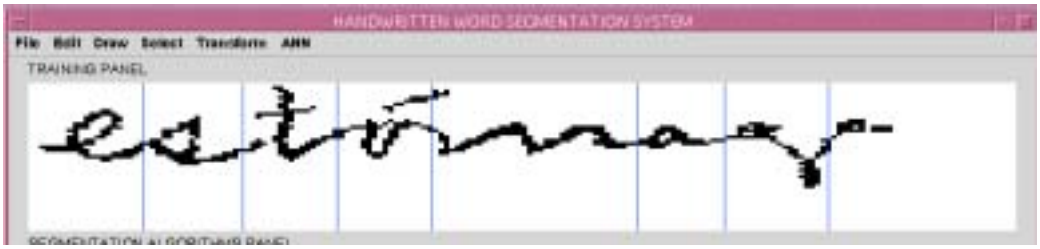
Numero de caracteres en la palabra = 5.

Figura 4.7 Algoritmo de histogramas

Así mismo se realizó un proceso de cálculo de promedio general de ancho de letra a través de un algoritmo semiautomático. Primero se realiza la lectura de la imagen, luego el usuario la observa y analiza y va realizando *clicks* sobre la palabra donde se cree existen puntos de segmentación. El algoritmo recibe las coordenadas de los *clicks* y saca el promedio de ancho de letra después de cada *click*. Este procedimiento se continúa enriqueciendo con un número considerable de imágenes hasta obtener un promedio

general de varias palabras. La figura 4.8 muestra la interfaz de la herramienta HAWOST (Hand Written Word Segmentation Tool) que fue desarrollada para realizar lo anterior.

a) Eleccion de la modalidad para dibujo de puntos de segmentacion



a) Puntos de segmentacion dibujados por el usuario

Tamaño original de la imagen en el eje x = 490 píxeles
Numero de caracteres encontrados (mediante algoritmo de histogramas) = 8
Promedio de ancho de caracter calculado por la herramienta = 40 píxeles
Nuevo tamaño de la palabra= 8*40= 320 pixeles

Figura 4.8 Cálculo de ancho de letra usando
la herramienta HAWOST

Por último el nuevo tamaño de la palabra estará determinado por la multiplicación del número de puntos de segmentación aproximados por el tamaño promedio de letra. Es necesario recalcar que este algoritmo no proporciona tamaños de

letra exactos, pero si proporciona una aproximación bastante beneficiosa. Además de brindar un método de normalización automático, HAWOST proporciona otras operaciones como la segmentación manual de una palabra, segmentación a través de un algoritmo predefinido o el uso de una RNA asesora. Estos puntos serán descritos en los capítulos 6 y 7.