UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

EXTRACCIÓN Y RECONOCIMIENTO DE RÓTULOS EN VÍDEOS MPEG

JUAN IGLESIAS CAVADA

MARZO 2009

EXTRACCIÓN Y RECONOCIMIENTO DE RÓTULOS EN VÍDEOS MPEG

Autor: Juan Iglesias Cavada

Tutor: Jesús Bescós Cano

Video Processing and Understanding Lab

Dpto. de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Marzo de 2009

Resumen:

En este proyecto se han implementado los algoritmos necesarios para segmentar texto superpuesto en imágenes o fotogramas de vídeo. Primero se estima el ancho del trazo de los caracteres a partir de la información obtenida de su perfil de proyección. Con esta información se genera un elemento estructurante adecuado para ejecutar la operación morfológica top-hat, que resalta el texto sobre el fondo. Posteriormente, se realiza una umbralización haciendo uso del método de Otsu y se efectúa un análisis de componentes conexas para, atendiendo a unas condiciones restrictivas, eliminar las zonas que son consideradas errores. Con el resultado obtenido, se alimenta un OCR, que reconoce el texto en la imagen y lo traduce a texto editable.

Abstract:

The algorithms described in this project achieve the segmentation of text embedded in images or video frames by means of morphological operations. First, the character's stroke thickness is estimated, and suitable structuring element is generated. Then morphological top-hat operation is performed in order to highlight the text from the background. Next, thresholding is carried out using Otsu's approach. Finally, connected component analysis is done and some areas are erased according to some restrictive conditions, so that errors are discarded. Then the resulting image is forwarded to OCR and recognition takes place.

Palabras clave:

Texto, extracción, segmentación, OCR, reconocimiento, caracteres, caption, captions, rótulos, MPEG.

Agradecimientos

Quiero agradecer a mi tutor, Jesús Bescós, el trabajo y el tiempo que ha dedicado a que este proyecto se haya podido realizar. A mis padres, por su incondicional apoyo y cariño. Y finalmente, a todos quienes han hecho posible que estos años hayan sido inolvidables.

Juan Iglesias Cavada.

Marzo 2009.

Índice

Resumen 4

Abstract 4

1 Introducción 1
1.1 Motivación y objetivos 2
1.2 Organización de la memoria 2
2 Estado del arte 4
2.1 Estado del arte 4
2.2 Resultados publicados 7
2.3 Conclusión 9
3 Descripción del sistema 11
3.1 Introducción 11
3.2 Algoritmo 13
3.2.1 Finalidad del algoritmo 13
3.2.2 Estimación del ancho del trazo de los caracteres 15
3.2.2.1 Estimación de la posición de una línea de texto 16
3.2.2.2 Estimación de la anchura de los caracteres 17
3.2.2.3 Métodos descartados 19
3.2.3 Tratamiento morfológico del texto: top-hat 21
3.2.4 Cambio de tamaño de la imagen 25
3.2.5 Binarización 26
3.2.6 Análisis de componentes conexas 27
3.2.7 Reconocimiento de caracteres: OCR 30
4 Diseño software e implementación 32
4.1 Especificaciones del sistema 32
4.2 Organización del programa: Clases y funciones 32
4.3 Fases del algoritmo 34
4.3.1 Lectura de la imagen 34
4.3.2 Decisión texto normal o inverso 34
4.3.3 Estimación del ancho de trazo de los caracteres 35
4.3.4 Top-hat de la imagen 38
4.3.5 Aumento de la resolución de la imagen 39
4.3.6 Umbralización 40
4.3.7 Análisis de componentes conexas 40
4.3.8 Reconocimiento de caracteres 43
4.4 Integración 45
5 Pruebas y resultados 47
5.1 Base de datos de prueba 47
5.2 Realización de las pruebas 47
5.2.1 Indicadores de calidad 47
5.2.2 Resultados 48

5.2.3 CONSIDERACIONES SOBRE EFICIENCIA 50
6 CONCLUSIONES Y TRABAJO FUTURO 51
6.1 CONCLUSIONES 51
6.2 TRABAJO FUTURO 51

REFERENCIAS I

GLOSARIO III

ANEXOS IV

A. MORPHOLOGICAL APPROACH TO CAPTION SEGMENTATION AND RECOGNITION IN VIDEO FRAMES .. IV
B. BASES DE DATOS UTILIZADAS IX

PRESUPUESTO XVI

PLIEGO DE CONDICIONES XVII

Índice de figuras

Figura 3.1.1 Estructura del IVOnLA 12

Figura 3.1.2. Esquema de marcado de las bounding boxes. 13

Figura 3.2.1. Diagrama de las etapas del sistema. 14

Figura 3.2.2 Etapas comprendidas en el algoritmo de segmentación. 15

Figura 3.2.3: Ejemplo de perfil de proyección horizontal 16

Figura 3.2.4. Imagen original e imagen filtrada con filtro detector de bordes de Canny. 17

Figura 3.2.5. Área sobre la que se computa el perfil horizontal para evitar errores (arriba) y perfil de proyección vertical de dicha área (abajo). 18

Figura 3.2.6. Perfil de proyección vertical (arriba) y el mismo perfil al que se ha restado su media y truncado (abajo). 19

Figura 3.2.7. Imagen y su respectiva transformada de Fourier (arriba) y corte transversal de la transformada (abajo). 20

Figura 3.2.8. Imagen y su respectiva transformada de Fourier (arriba) y corte transversal de la transformada (abajo). 21

Figura 3.2.9. Ejemplo de elemento estructurante cuadrado de cinco por cinco 22

Figura 3.2.10. Elemento estructurante de forma romboidal. 22

Figura 3.2.11. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas. 23

Figura 3.2.12. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas. 24

Figura 3.2.13. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas. 24

Figura 3.2.14 Distribución de densidad de los 26 caracteres del alfabeto inglés, de fuente arial regular, con diferentes tamaños entre 6 y 22 puntos. Imagen tomada de [9]. 30

Figura 4.3.1 Rótulo filtrado con un filtro de bordes Canny, y con errores en el fondo. 37

FIGURA 4.3.2 PERFIL DE PROYECCIÓN HORIZONTAL. EN ROJO, EL NIVEL MEDIO. 37

FIGURA 4.3.3 ELEMENTO ESTRUCTURANTE ROMBOIDAL DE CINCO POR CINCO PÍXELES 38

FIGURA 4.3.4 IMAGEN UMBRALIZADA ANTES DEL ANÁLISIS DE COMPONENTES CONEXAS. 43

FIGURA 4.3.5 IMAGEN UMBRALIZADA DESPUÉS DEL FILTRADO DE COMPONENTES Y BINARIZADO FINAL CON CARACTERES NEGROS Y FONDO BLANCO. 43

FIGURA 5.2.1 EJEMPLOS DE RESULTADOS SOBRE RÓTULOS DE PRUEBA. 50

Índice de tablas

TABLA 2.1. RESULTADOS PUBLICADOS POR [5] 8

TABLA 2.2. RESULTADOS PUBLICADOS POR [1] 8

TABLA 2.3. RESULTADOS PUBLICADOS POR [2] 9

TABLA 2.4. RESULTADOS PUBLICADOS POR [3] 9

TABLA 2.5. RESULTADOS PUBLICADOS POR [6] 9

1 Introducción

El creciente desarrollo de las tecnologías multimedia ha supuesto recientemente un cambio en la forma de transmitir y recibir información. Ingentes cantidades de datos son puestos diariamente a disposición de usuarios de todo el mundo, por medio de diversos canales como internet, televisión digital, satélites etc. Esta explosión trae consigo la necesidad inherente de localización y clasificación de la información, es decir, el usuario debe ser capaz de discernir los contenidos de su interés entre toda esa vasta maraña de datos. Es por esto que no se puede dejar de tener en cuenta ninguna de las opciones que permiten o facilitan labores de indexación, clasificación, resumen o localización de datos. De entre ellas, este proyecto se centra en la información textual presentada en forma de rótulos o *captions*, que son textos breves superpuestos en una imagen o vídeo y que contienen información explicativa acerca de lo que hay de relevante en dicha imagen/vídeo.

Un sistema de este tipo suele constar de una sucesión de tareas diferenciadas que son requeridas para lograr la correcta extracción del texto, a saber:

-***Localización de fotogramas con texto***: De entre todos los fotogramas de que consta un vídeo, es importante localizar los que contienen información textual superpuesta, además de la situación precisa del texto dentro de un fotograma. Esto es imprescindible para el buen funcionamiento de un posterior algoritmo de *segmentación* o extracción de una máscara binaria con el texto. No se considera en este proyecto la implementación de esta etapa, puesto que se dispone de un programa que la lleva a cabo, desarrollado dentro del Grupo de Investigación.

-***Segmentación del texto***: Es el núcleo del trabajo que se ha efectuado en este proyecto, consistente en la aplicación de diversos algoritmos que, partiendo de una imagen (una región de un fotograma) en la que se ha localizado previamente información textual, separan los caracteres del fondo, obteniéndose finalmente otra imagen binaria en la que el fondo es blanco y los caracteres negros. Dicho formato es imprescindible para el correcto desarrollo de la siguiente y última etapa.

-***Reconocimiento***: En una imagen en la que aparece información textual, ésta forma parte del contenido de dicha imagen, es decir, está compuesta por píxeles. Los caracteres se representan como un conjunto de puntos del mismo color, textura o cualquier otra característica que permita identificarlo como tal. En cambio, un ordenador almacena los caracteres como códigos, lo cual permite llevar a cabo tareas como la búsqueda, la edición, e incluso el cambio de tamaño o tipo de letra. El objetivo que se persigue es la traducción desde el primer tipo de datos al segundo. Esta etapa es llevada a cabo por un programa diseñado al efecto denominado OCR (*Optical*

Character Recognition), programa cuya entrada ha de ser una imagen binaria, o al menos bimodal.

Después de pasar por estas tres etapas, se ha logrado obtener el texto que se encontraba incrustado en una imagen, permitiendo efectuar búsquedas y localizar la información dentro del vídeo.

Se han dedicado grandes esfuerzos a desarrollar aplicaciones con el fin de realizar estas tareas, no obstante, es un asunto que aún no ha sido completamente resuelto por los investigadores, debido a la gran cantidad de formatos que pueden presentar los rótulos, así como a las dificultades añadidas, como la baja calidad. Por tanto, se siguen presentando alternativas y métodos novedosos constantemente, lo que otorga a este proyecto un carácter altamente investigador.

1.1 Motivación y Objetivos

Las razones que motivan el desarrollo de un proyecto de estas características son claras, ya que posibilita un ahorro en el tiempo invertido para localizar vídeos con determinada información, siempre que ésta venga acompañada de rótulos (lo cual es muy común en noticiarios y programas de actualidad) así como en la indexación de vídeos dentro de grandes bases de datos.

El objetivo principal de este proyecto es el desarrollo de un algoritmo que logre procesar una imagen con texto hasta generar los códigos de caracteres que representa, tal como se ha visto. Para ello, se deben completar exitosamente los siguientes puntos, partiendo de una imagen que contenga un rótulo con información textual:

- Generación de algoritmos que posibiliten la separación de los caracteres contenidos en dicho rótulo del fondo que los sustenta y demás elementos perturbadores (ruido, errores, etc.).
- Alimentación de un programa de reconocimiento óptico de caracteres (*OCR*) con el resultado de la etapa previa, y recogida de la salida (es decir, del texto).
- Integración de todas las etapas en un único algoritmo.

1.2 Organización de la memoria

La memoria consta de los siguientes capítulos:

- ***Capítulo 1:*** Introducción, motivación y objetivos del proyecto.

- ***Capítulo 2***: Repaso del estado del arte en esta materia, comentando las técnicas más comúnmente utilizadas en este ámbito y los resultados obtenidos por otros investigadores en tareas similares.
- ***Capítulo 3***: Descripción del sistema, justificación y explicación de todas las etapas y operaciones que realiza el algoritmo.
- ***Capítulo 4***: Organización del programa, detalles técnicos del sistema, organización de clases y funciones. Justificación de cómo se han programado las diferentes etapas del algoritmo.
- ***Capítulo 5***: Resultados experimentales generados por el sistema. Descripción de la batería de datos de prueba utilizada.
- ***Capítulo 6***: Conclusiones observadas al término del proyecto. Propuestas sobre líneas interesantes de trabajo futuro.

2 Estado del arte

2.1 Estado del arte

Varios son los métodos aplicables a la tarea de segmentación de texto en imágenes con fondo complejo. Difieren tanto en las características explotadas como en la finalidad perseguida. Así, podemos diferenciar los que se centran en la extracción de texto en imágenes de la vida real(p. e. fotografías), los que utilizan características propias de las secuencias de vídeo, los que se especializan en imágenes en páginas web, o aquellos cuya finalidad es obtener texto artificialmente superpuesto en imágenes, como es este caso. La focalización permite obtener mejores resultados en un campo específico.

Los algoritmos de extracción de texto en imágenes van a menudo precedidos por otros llamados de localización. Es muy útil disponer de unas coordenadas dentro de la imagen que esclarezcan con la mayor precisión posible dónde se encuentra el texto. Las particularidades que posee una imagen con texto y que hacen que pueda ser extraído y reconocido sólo se observan en un entorno pequeño del texto que se denomina *bounding box*. Las *bounding boxes* son cajas rectangulares que contienen al texto, y que se ajustan a éste, es decir, no son mucho más grandes que la región de texto. Con objeto de localizarlas y delimitarlas, se han desarrollado diversos métodos cuyo análisis no corresponde a este proyecto.

La cascada de acciones que se llevan a cabo es siempre localización, segmentación y reconocimiento. Es la etapa segunda la que se analiza en este proyecto, ya que el reconocimiento se hace de mano de un tipo de programas llamados OCR (*Optical Character Recognition*).

Veamos a continuación un resumen de los métodos más empleados y ejemplos de cómo los investigadores han lidiado con el problema de la segmentación de texto.

El color del texto es una de las características utilizables, ya que la legibilidad de éste le obliga a diferir en gran medida del fondo que lo sustenta. Según esto, algunos autores dedican esfuerzos a estimar el color del fondo y del texto, como Julinda Gllavata [1] o Lienhart [3] mediante el método de resta de histogramas en diferentes zonas de la imagen, o bien localizando zonas pertenecientes al interior de las letras para obtener un perfil claro del color del texto y lograr así aislarlo, como en [2]. Se ha profundizado en este campo mediante el uso de diferentes espacios de color aparte del cásico RGB ([1],[3]). Como ejemplo, Qixiang Ye et al [2], que utiliza el espacio HSV (aunque sólo las componentes H y S) para entrenar GMM's (*Gaussian Mixture Models*) con el color de píxeles pertenecientes al texto, para luego decidir mediante dichos modelos si un píxel

cualquiera de la imagen forma parte o no del texto. Utilizando un filtro de bordes, encuentran pares de líneas horizontales y verticales que pertenecen a las letras. Con esta información, muestrean píxeles que pertenecen con gran probabilidad al interior de una letra, creando así un modelo de color que les permite, comparándolo con un píxel cualquiera de la imagen decidir si se considera texto o fondo.

Incluso se han usado, como hace Karatzas en [9], espacios como el perceptualmente uniforme (L*a*b*), en el que la distancia euclídea entre colores es proporcional a variaciones en la percepción visual humana, característica no observable en otros espacios de color más comunes como RGB o HSV.

Los métodos de clasificación o *clustering* gozan de gran aceptación. Consisten en generar varias clases o *clusters*, y posteriormente ir asignando los píxeles a ellas. En [10], García et al, separan los 4 colores más dominantes de la imagen mediante el algoritmo *k-means*, asignándole a cada píxel el color en medio del segmento correspondiente. Tras esto, se eliminan de toda la imagen uno o dos colores según su presencia en un entorno exterior a los límites de la *bounding box*, ya que se suponen pertenecientes al fondo. Es decir, se toma una *bounding box* ligeramente mayor que la original, y en la zona que queda entre ambas se estudia cuál de los 4 colores que se habían aislado antes se encuentra más presente. Este color es considerado fondo y por tanto se elimina de toda la *bounding box*. Si hay algún otro color con una aparición suficiente en esta zona se elimina también, apoyándose en la misma hipótesis. Con los restantes, conjeturan que una combinación de ellos es el resultado óptimo, esto es, alguna de las imágenes que se forman tomando uno o varios de los colores supervivientes tiene que contener al texto, siendo el fondo lo descartado. Para verificar cuál de las posibles combinaciones es la correcta, estudian las propiedades de periodicidad de los trazos de los caracteres, tomando la imagen en una dimensión (horizontalmente). Se apoyan para ello en la particularidad de que los trazos y los espacios entre letras mantienen un ancho uniforme. Al probar este sistema encontramos que, ocasionalmente, los 4 colores mayoritarios pertenecen al fondo, por lo que el texto se verá integrado en alguna de las zonas de color, haciendo imposible su diferenciación.

También en [1], tras la estimación de colores, se usa *k-means* para clasificar los píxeles usando como vector de parámetros los valores RGB y la desviación estándar de los coeficientes wavelet para considerar las características locales del texto.

En este campo es también común encontrarse con esquemas de tipo *split and merge* (subdividir y agregar). En [5], Yaowen Zhan et al, implementan un modelo de esta forma. En la etapa de *split*, llevan a cabo un análisis de componentes conectados, para posteriormente eliminar aquellos que linden con el borde de la *bounding box*, aduciendo que el texto se distribuye en la parte central de la misma. Para eliminar el resto de componentes sobrantes, definen un parámetro al que denominan escala de

un componente. Para cada píxel en el interior de un componente conectado, acumulan la distancia más corta de todos los segmentos que unen bordes dentro del componente y pasan por ese píxel, en 4 direcciones (horizontal, vertical y diagonal) y dividen por el número de píxeles. Si un componente presenta una escala demasiado grande es eliminado. En la etapa de agregación, se generan de forma adaptativa varias clases, y se asignan los píxeles mediante el algoritmo *k-means*, utilizando para los vectores de características la escala del componente y el color medio de los píxeles que lo forman. El número de clases crece hasta que la varianza de los *clusters* rebasa un umbral definido. Cada clase resulta en una capa, supuesta una de éstas conteniendo al texto. Para seleccionar la capa válida, se someten todas a unas sencillas reglas referidas al número total de píxeles en la capa, número de componentes conectados y a ratios entre áreas de componentes, etc. En una etapa de pos procesado, y una vez decidida la capa conteniendo al texto, se decide si incluir algunos píxeles que se hayan podido perder en base a unas reglas de vecindad y color. Con el resultado se puede alimentar un OCR. Este esquema se puede encontrar también en [9].

En [5], al igual que en [3], se usa un algoritmo de crecimiento (*SeedFill*) para encontrar los componentes conectados. A diferencia del método de etiquetado, en el que se asigna una etiqueta a los píxeles según el componente al que pertenezcan, este algoritmo asigna un color común a los píxeles cercanos cuyo color no difiera en más de un umbral del color de la 'semilla'. Ambos autores asumen que los bordes de la *bounding box*, están fuera de los límites del texto en la imagen, por tanto, píxeles del color de alguno de los bordes y conectados con éstos, según el algoritmo de la semilla, se eliminan. Para ello, se extienden las *bounding boxes* en un número de píxeles. Este sistema puede acarrear problemas en caso de líneas de texto demasiado próximas, y además eliminará cualquier letra que por error esté en contacto con los límites. Por demás, el coste computacional de aplicar este algoritmo es alto, y no apropiado para sistemas de tiempo real.

El uso de componentes conectados tiene otra aplicación profusamente explotada, ya que permite descartar zonas del resultado final que se hayan incluido erróneamente. Es común que, después de todo el desarrollo del proceso, se encuentren manchas no correspondientes con caracteres, que por sus características no hayan sido eliminadas. Habitualmente, es posible descartar gran parte de ellas en base a sus propiedades morfológicas, es decir, que sean demasiado largas o demasiado grandes y viceversa. Por ejemplo, Chen ([6]), aplica restricciones en cuanto al número de píxeles, al ratio alto/ancho, y al ratio ancho/alto total de la región en cuestión, eliminándola si no cumple los requisitos. Podemos ver esquemas similares de posprocesado en [2][3][4][5] y otros.

Algunos autores descartan la información de color argumentando que aún en niveles de gris el texto ha de estar claramente diferenciado. Existen algunos algoritmos útiles para encontrar un umbral de binarización óptimo. Por ejemplo, Jovanka Malobabic ([8]), asume que la distribución estadística de píxeles tanto de fondo como de texto es gaussiana, y los asigna de forma que se maximice la varianza entre clases. Otras posibilidades son la manipulación de histogramas hasta lograr que se conviertan en bimodales , como en [7], o bien la subdivisión progresiva de la imagen hasta que sólo comprenda dos zonas diferenciadas.

En lo tocante al vídeo, en ocasiones puede ser útil la integración de varios fotogramas o *frames*. La ventaja de éste sistema radica en que, mientras el fondo cambia, el texto es fijo, por lo que se puede segmentar basándose en las zonas no variables de la integración de un número de fotogramas. No obstante, ésta modalidad no ofrece por sí sola gran utilidad, ya que no siempre todo el fondo cambia durante la aparición del rótulo, e incluso en muchas ocasiones se coloca un fondo superpuesto detrás para impedir que partes de la imagen dificulten la lectura del texto.

Otra de las técnicas habituales es el uso de múltiples hipótesis. El procedimiento se fundamenta en generar más de una salida y observar a posteriori la validez de alguna de ellas. Chen, en [6], produce distintas segmentaciones separando la imagen original en varias capas, según un factor k. Se agrupan los píxeles en base a su nivel de gris. Por ejemplo, para k=2 se componen 2 subimágenes, cada una con los píxeles que van hasta y desde un valor de intensidad concreto. Para un k=4, crea divisiones en 2, 3 y 4 capas, y las segmenta usando algoritmos como EM (*Expectation-Maximization*) o *Gibbsian EM*, con lo que obtiene 2+3+4=9 hipótesis, siendo una de ellas la que contiene el texto. A continuación se hace análisis de componentes conectados y se eliminan las regiones que no cumplan condiciones de tamaño y forma. Para mejorar el resultado de esta fase, se crea, para las zonas restantes, una restricción de consistencia en el nivel de gris, que mejora el filtrado de errores. Componentes que no la cumplan son asimismo eliminados. Seguidamente se transfieren las diferentes capas a un OCR (*Optical Character Recognition*), decidiendo según el resultado cuál de entre ellas es más óptima. Semejantemente, García et al en [10] examinan múltiples posibilidades en función de una segmentación por colores, tal como se ha descrito anteriormente.

El uso de operadores morfológicos en segmentación de caracteres ha sido pobremente desarrollado. A pesar de que se pueden hallar ejemplos en la literatura, este tipo de práctica ha sido tradicionalmente empleada para eliminación de ruido, creación de puentes perdidos entre fragmentos de un mismo carácter u otras finalidades, como se puede ver en [8], en que se sirven de ello para delimitar las *bounding boxes*.

2.2 Resultados publicados

No resulta sencillo elaborar un buen balance de resultados en lo referente a este asunto. Los motivos principales son la falta de bases de datos estandarizadas, por una parte, y por otra la diversidad de objetivos que se pueden perseguir. Por ello, un tipo concreto de técnica sólo resultará en imágenes de cierta índole. Asimismo, no ha sido probada la relevancia del uso de un *OCR* concreto en detrimento de otros. La influencia de éste podría llegar a ser notable.

Los parámetros de calidad usualmente utilizados para medir la bondad de un sistema de estas características tienen que ver con la eficacia a la hora de segmentar y reconocer los caracteres. Así, un sistema es mejor cuantos más caracteres es capaz de segmentar y reconocer. Por ello, las medidas más frecuentemente expresadas son la tasa de reconocimiento de carácter (*Character Recognition Rate*) y la tasa de reconocimiento de palabra (*Word Recognition Rate*). Una palabra se considera correctamente reconocida cuando todos sus caracteres han sido bien reconocidos, ya que en un sistema real, un fallo en un solo carácter impediría localizar la palabra en cuestión.

Se presentan a continuación algunos resultados de los métodos expuestos publicados por sus autores:

[5]: La tabla siguiente muestra los resultados obtenidos por Yaowen Zhan et al., que disponían de 250 imágenes recogidas de internet y 350 fotogramas de vídeo.

	Bloques de texto	CER	CRR
Chino	355	96.5%	82.4%
Inglés	442	93.6%	84.8%
Mixto	797	94.3%	84.2%

Tabla 2.1. Resultados publicados por [5]

CRR: *Character Recognition Rate*, tasa de reconocimiento de carácter.

CER: *Character Extraction Rate*, tasa de extracción de carácter.

[1]: Resultados obtenidos por Julinda Gllavata et al., utilizando un set de prueba de 38 imágenes seleccionadas de entre imágenes tomadas de televisión, páginas web y noticiarios con una amplia diversidad de fondos y tipos de texto, y fotogramas extraídos de vídeos MPEG-7, conteniendo en total 325 palabras y 2156 caracteres. El OCR seleccionado para reconocimiento es ABBYY FineReader 7.0 Professional.

Tasa de reconocimiento de palabra	Tasa de reconocimiento de carácter
55.83%	77.71%

Tabla 2.2. Resultados publicados por [1]

[2]: Resultados obtenidos por Qixiang Ye et al., que usaron un conjunto de prueba compuesto por 100 imágenes de páginas web y 300 fotogramas de vídeo, en los que había 4861 palabras chinas.

Tasa de reconocimiento de palabra	Velocidad (caracteres/segundo)
85.5%	252

Tabla 2.3. Resultados publicados por [2]

[3]: Resultados de Lienhart y Wernikle, cuyo set de prueba consiste en veintitrés secuencias de vídeo, tanto doméstico como retransmitido, abarcando texto con diferentes características, al que se añadieron además imágenes pertenecientes a siete páginas web, conteniendo 2187 caracteres. Definen la tasa de extracción de carácter como el porcentaje de caracteres correctamente segmentados, esto es, sin errores evidentes en su forma.

CER	CRR
79.6%	69.9%

Tabla 2.4. Resultados publicados por [3]

CER: Tasa de extracción de carácter (*Character Extraction Rate*).

CRR: Tasa de reconocimiento de carácter (*Character Recognition Rate*).

[6]: Resultados de Chen. Su conjunto de datos de prueba incluye 1208 imágenes seleccionadas de vídeos deportivos de la BBC, con 9562 caracteres y 867 palabras. Los resultados se muestran para distinto número de hipótesis (k) y para los distintos algoritmos de binarización (EM, GEM o K-MEANS).

K	Algoritmo	Extraidos	CRR	CPR	WRR
2,3	EM	9449	94.0%	95.3%	88.1%
2,3	GEM	9579	96.5%	96.5%	92.8%
2,3	K-MEANS	9587	97.1%	97.0%	93.7%
2,3,4	EM	9411	93.9%	95.6%	88.1%
2,3,4	GEM	9557	96.6%	96.8%	93.0%
2,3,4	K-MEANS	9560	97.0%	97.2%	93.8%

Tabla 2.5. Resultados publicados por [6]

CPR: *Character Precision Rate*. Tasa de precisión de carácter (reconocidos/extraídos).

WRR: *Word Recognition Rate*. Tasa de reconocimiento de palabra.

2.3 Conclusión

Tras examinar las distintas hipótesis planteadas, se puede deducir que no existe actualmente una técnica que supere claramente al resto. Los investigadores siguen dedicando esfuerzos a desarrollar métodos eficientes para segmentación de texto. Podríamos decir simplemente que, aunque la tarea sea común, la amplia diversidad de tipos de texto en imágenes hace que no sea factible la aplicación de un solo algoritmo. Es, por tanto, responsabilidad del usuario delimitar las características de sus necesidades en aras de seleccionar un procedimiento adecuado.

3 Descripción del sistema

3.1 Introducción

La adquisición de información textual perteneciente a imágenes o vídeos puede proporcionar, como hemos visto, un valioso método de indexación que, debido a la saturación de fuentes multimedia disponibles, es importante aprovechar. El objetivo inicial perseguido en este proyecto es, a partir de rótulos superpuestos a una secuencia de vídeo, la extracción o separación de los caracteres que los componen del fondo que los envuelve, obteniendo así una imagen binaria. Una vez llevada a cabo esta tarea, un software OCR (*Optical Character Recognition*) se encargará de transformar dicha imagen binaria de caracteres en los códigos correspondientes, de forma que puedan ser almacenados y, así, objeto de búsqueda en una base de datos. De esta forma es sencillo localizar secuencias de vídeo cuyo contenido responde a unas pocas palabras clave. Por ejemplo, si en los informativos de un año se desea localizar las noticias que trataban sobre terremotos, la búsqueda por esa palabra clave devolverá la información necesaria para localizar la(s) noticia(s), como por ejemplo mes, día y minuto en que se emitió. Para ello se han implementado una serie de etapas que, consecutivamente, van aislando los caracteres del fondo que los sustenta. Dichas etapas serán detalladas en el punto siguiente.

Usualmente, no todos los fotogramas de un vídeo muestran información contextual. Por ejemplo, en un telediario es común verlos al comienzo de cada noticia, y quizás también cuando aparece algún personaje relevante. Si se pretende aislar el texto de estos rótulos, es preciso localizarlos previamente. En este sentido, el grupo de investigación dentro del cual se ha desarrollado este proyecto disponía ya de un sistema de localización tanto de los *frames* con información textual como de la delimitación de la *bounding box* que la contiene (es decir, del mínimo área rectangular que la delimita).

La concatenación de estos dos subsistemas se realiza dentro de un sistema mayor, un proyecto cuya finalidad es analizar vídeos y extraer de ellos la máxima información posible. Este sistema se denomina *IVOnLA [12]*.

El *IVOnLA* (*Image & Video ONline-Analysis module*) es un módulo encargado de la extracción en tiempo real de descriptores visuales de secuencias de vídeo de entre las consideradas en el sistema *MESH* (*http://www.mesh-ip.eu/?Page=Project*), esencialmente noticiarios. Estos descriptores sirven para la obtención de información relacionada con las noticias, según van apareciendo en el repositorio de contenido del *MESH*, y llevar a cabo las tareas de clasificación y resumen en tiempo real.

IVOnLA ha sido creado para la operación en tiempo real, tarea que se ataja en dos sentidos. Primero se emplea un análisis *on line* imprescindible para poder operar conforme se reciben los medios. Los resultados del análisis se obtienen constantemente por medio de un esquema causal; los resultados correspondientes a un fotograma sólo dependen de los fotogramas anteriores. Por otro lado, se aplican técnicas de análisis en el dominio comprimido, aumentando la eficiencia mediante la obtención de información ya codificada en el *bit stream*.

La siguiente figura muestra la arquitectura del IVOnLA (figura 3.1). Las entradas posibles son vídeos comprimidos en formato MPEG-2 y eventuales parámetros para controlar la función de cada módulo. La salida es una descripción del contenido del vídeo, que incluye su estructura temporal, descriptores de movimiento global o de cámara, máscaras de segmentación basadas en movimiento, imágenes DC, coeficientes DCT e información contextual.

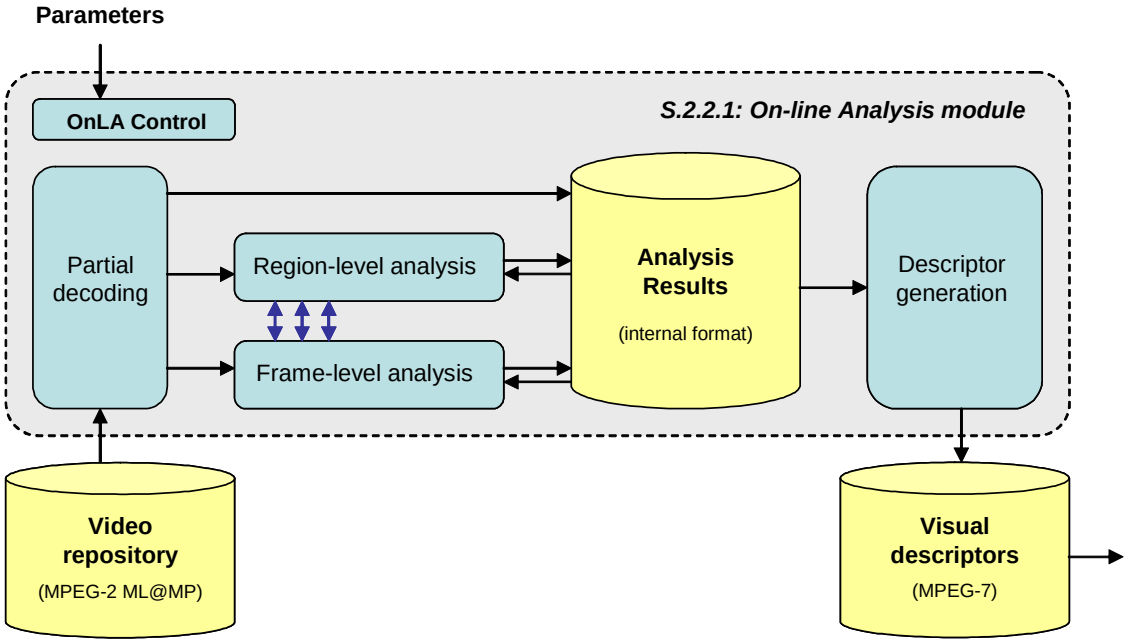


Figura 3.1.1 Estructura del IVOnLA

En lo que respecta a la extracción de contenido textual, primero se examina el vídeo en busca de *frames* con rótulos o *captions*, y después de localizar y marcar las *bounding b*oxes (ver figura 3.2), se procede a segmentar y reconocer el texto.

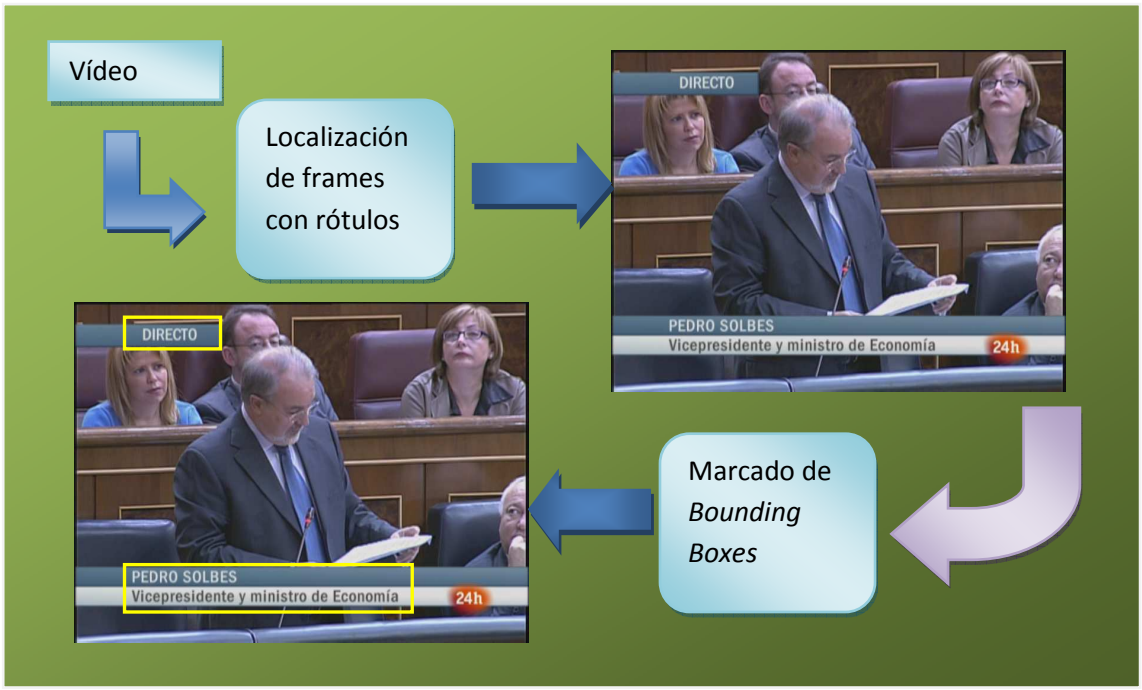


Figura 3.1.2. Esquema de marcado de las *bounding boxes*.

3.2 Algoritmo

3.2.1 Finalidad del algoritmo

Para que la información textual sea útil para la indexación de vídeo, se necesita disponer del texto que se encuentra incrustado en los rótulos, no sólo de su localización. La forma de pasar de un texto en una imagen (formado por píxeles) a un texto tratable como tal por un ordenador (formado por códigos de caracteres) es un programa diseñado al efecto, que se llama *OCR*.

No obstante, mientras que los *OCR's* están pensados para reconocer caracteres en imágenes binarias y de alta calidad, sencillas de obtener mediante el escaneado de libros o documentos con un formato sencillo, es común encontrar rótulos sobre fondos llamativos o complejos en los vídeos o televisión. Es por esto que para poder obtener un resultado global aceptable, se necesita tratar el texto hasta que se muestren caracteres negros sobre un fondo blanco y con el mínimo número de errores posible. A esta etapa se le denomina segmentación del texto (ver figura 3.2.1).

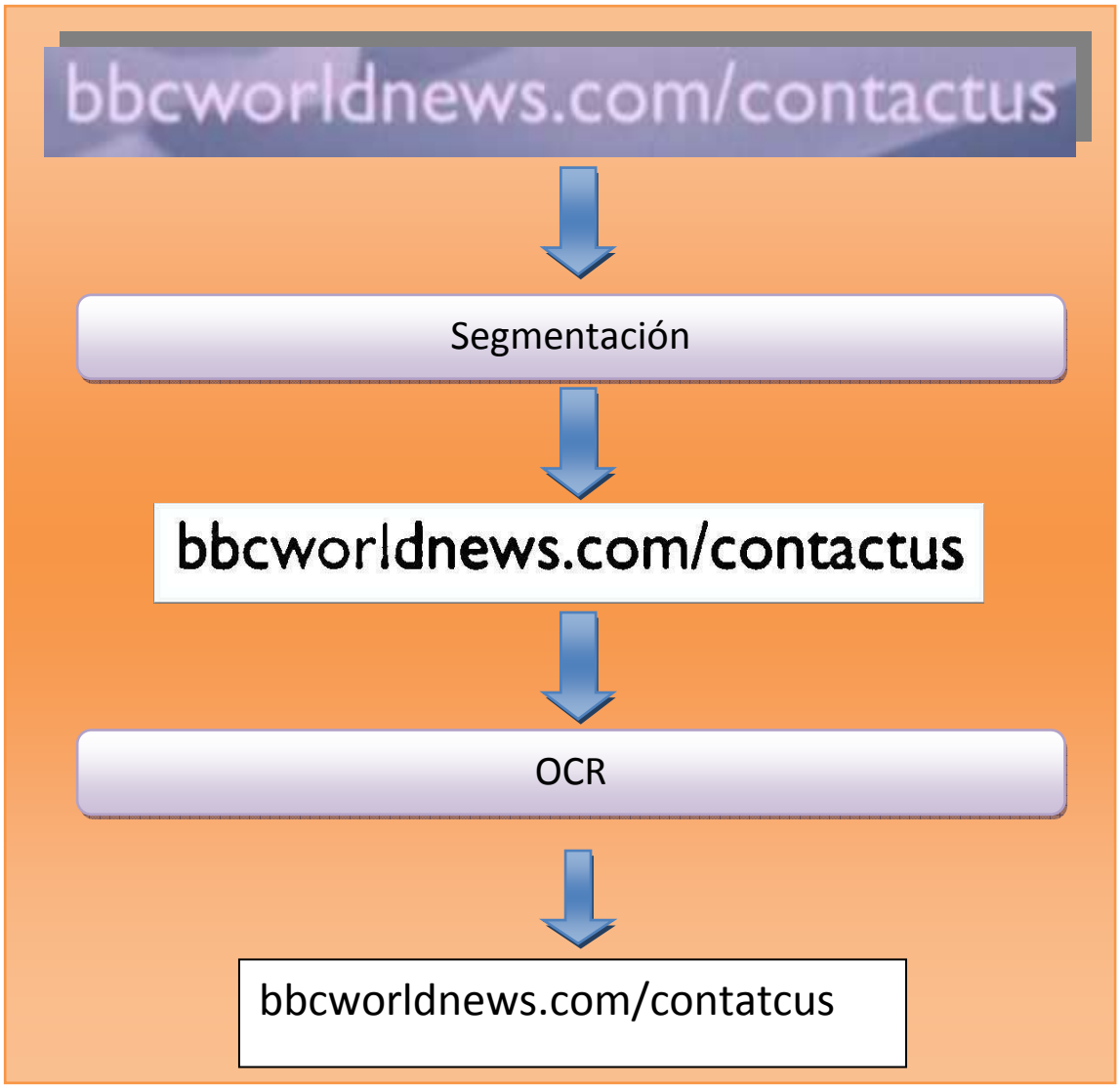


Figura 3.2.1. Diagrama de las etapas del sistema.

Dentro de la etapa de segmentación se pueden diferenciar, a su vez, distintos pasos que se han de efectuar sucesivamente para lograr la separación de texto y fondo (ver figura 3.2.2).

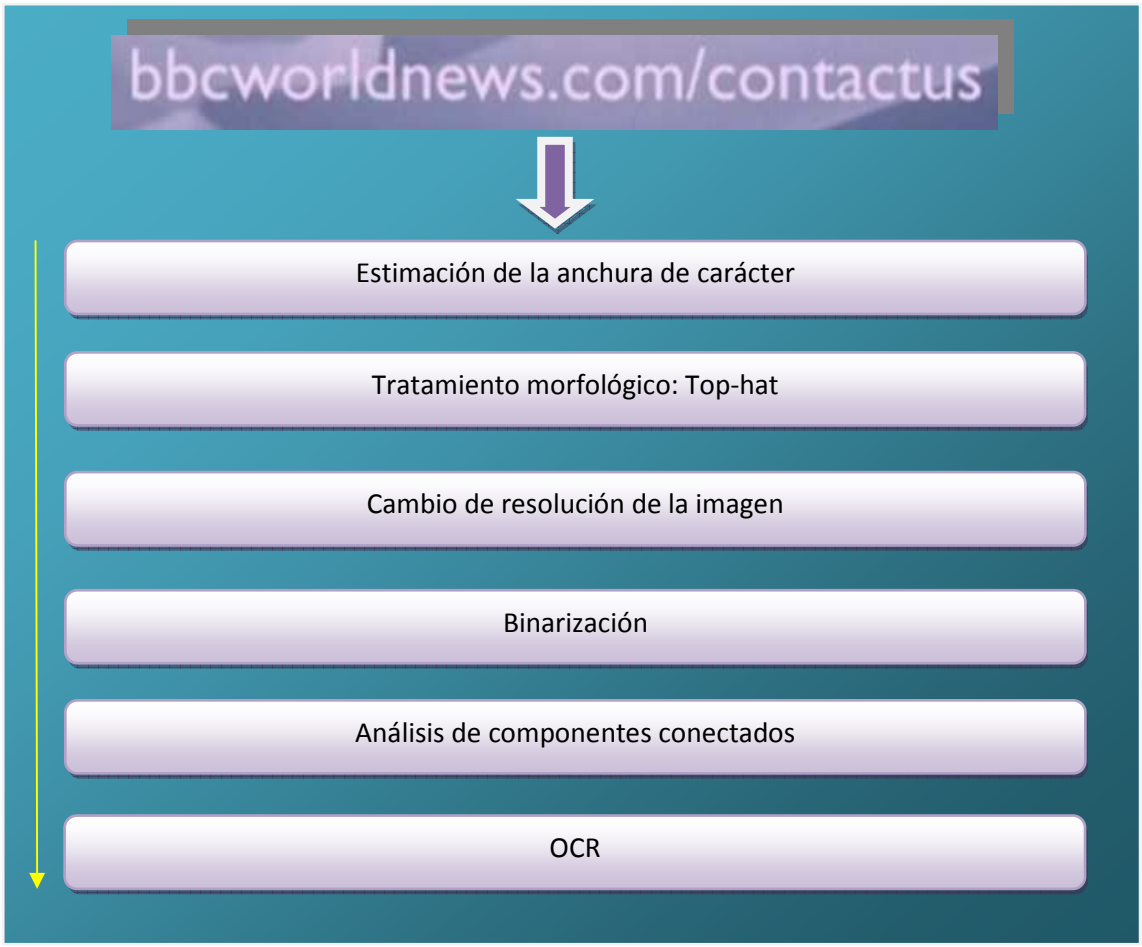


Figura 3.2.2 Etapas comprendidas en el algoritmo de segmentación.

Todos estos pasos le llevan a cabo sobre una imagen en niveles de gris. La información de color se descarta porque su procesado es considerablemente más lento que en una imagen en niveles de gris y porque la inherente necesidad de este tipo de texto de ser sencillo de leer asegura suficiente contraste entre fondo y caracteres como para poder ser segmentados únicamente con la información de intensidad.

3.2.2 Estimación del ancho de trazo de carácter

Dado que el sistema de segmentación de texto propuesto se fundamenta en tratamiento morfológico de imágenes, es absolutamente determinante obtener con precisión el tamaño de aquello que se pretende extraer, en este caso, el ancho del trazo de los caracteres.

Para llevar a cabo esta operación, se hace uso de lo que se denomina perfil de proyección (*Projection Profile*) de una imagen. El perfil de proyección es el valor acumulado de los píxeles de una imagen en una de sus direcciones principales (horizontal o vertical).

Para encontrar el ancho de trazo mediante el estudio del perfil de proyección vertical se analiza el ancho de los picos de dicho perfil, correspondientes a los trazos largos verticales de las letras. Por este motivo, el perfil de proyección debe computarse sólo en la zona de la imagen constituida por una única línea de texto, ya que la presencia de elementos del fondo o bien de caracteres pertenecientes a otras líneas introduciría distorsiones (en forma de picos) en este perfil impidiendo o dificultando el aislamiento de los picos que realmente permiten la obtención del ancho de carácter.

3.2.2.1 Estimación de la posición de una línea de texto

Las *bounding boxes* que encierran el texto se suponen cercanas a él. No obstante, para lograr el objetivo con precisión es necesario determinar fielmente las filas en las que empieza y termina una línea de texto, ya que si se procesase toda la *bouding box*, se introducirían en el cálculo elementos distorsionantes pertenecientes al fondo o incluso a otras líneas de texto pertenecientes al mismo rótulo.

Para localizar, pues, una línea de texto dentro de la *bounding box*, se recurre al perfil horizontal de proyección (ver fig. 3.2.3). En vez de computar este perfil sobre la imagen original, el cálculo se lleva a cabo sobre una imagen resultado de filtrar la inicial con un filtro detector de bordes de *Canny* (ver fig.3.2.4). Esto es debido a que el fondo puede contener elementos que alteren sensiblemente el perfil de proyección por intensidad, siendo más improbable que dichos elementos introduzcan grandes errores en el número de bordes.

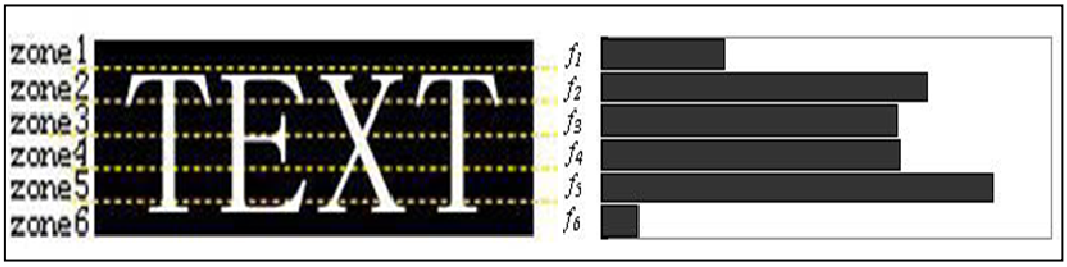


Figura 3.2.3: Ejemplo de perfil de proyección horizontal

Figura 3.2.4. Imagen original e imagen filtrada con filtro detector de bordes de Canny.

Como se aprecia en la figura 3.2.3, el perfil horizontal de proyección es alto en filas comprendidas en líneas de texto y es casi nulo en el resto de la imagen. Un entorno de filas con gran densidad relativa de bordes se supone, pues, perteneciente a una línea de texto. Posteriormente, se localizan a partir de estas filas centrales el inicio y el fin de la línea donde la densidad de bordes por fila sufre un descenso brusco.

3.2.2.2 Estimación de la anchura de los caracteres.

A continuación, se estima el ancho de carácter basándose en el perfil de proyección vertical de la imagen sin filtrar y delimitada por las filas inicial y final que se obtuvieron como resultado en el paso anterior. De esta forma, lograremos aislar el ancho de los trazos de las letras observando la anchura de los picos que se generan en el perfil de proyección, tal como se ve en la figura 3.2.5. La eficacia de este método reside en la mayor abundancia de trazos verticales respecto a los horizontales en la mayoría de los caracteres del alfabeto latino.

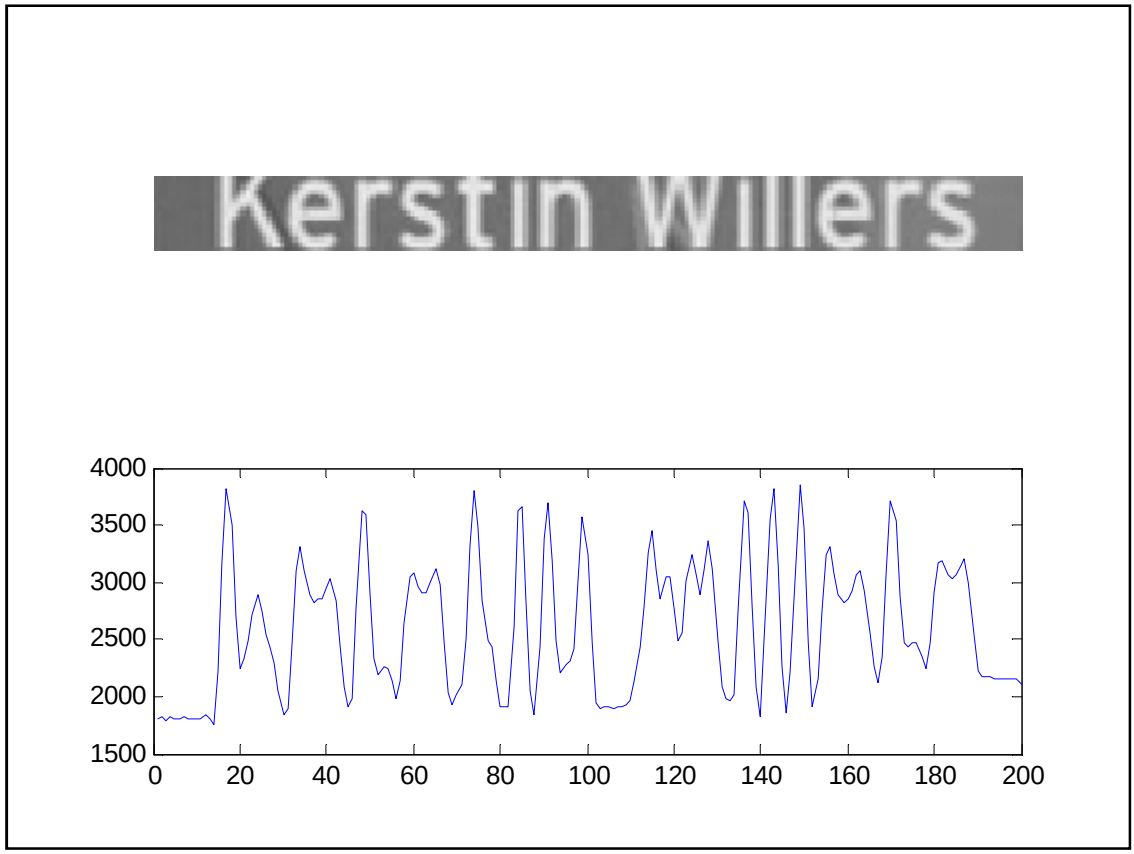


Figura 3.2.5. Área sobre la que se computa el perfil horizontal para evitar errores (arriba) y perfil de proyección vertical de dicha área (abajo).

Tal como se aprecia en el ejemplo, continúa siendo complicado estimar el ancho de trazo a partir del simple perfil vertical de la línea de texto, por lo que para facilitar la tarea se le resta su propia media, con el fin de aislar los picos de intensidad.

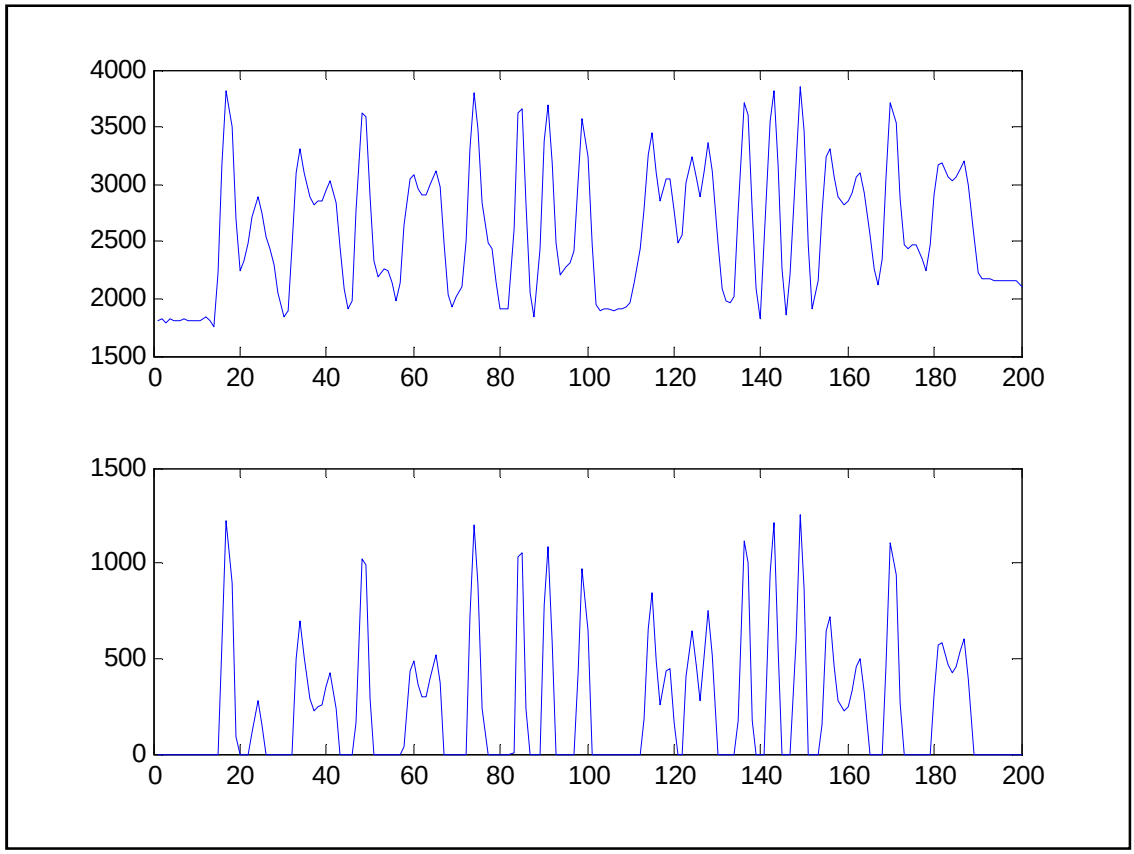


Figura 3.2.6. Perfil de proyección vertical (arriba) y el mismo perfil al que se ha restado su media y truncado (abajo).

De nuevo, se presenta un problema en la estimación, ya que no todos los caracteres muestran perfiles con picos aislados, apropiados para la estimación de anchura. Por ello, nos apoyamos en un estimador robusto de media, eliminando los *outliers*. En un conjunto de muestras, un *outlier* es una de dichas muestras que difiere en gran medida del resto y cuyo cómputo puede desplazar las características estadísticas respecto de las deseadas. Existen varios métodos que tienen como objetivo el cálculo de dichas características, descartando los *outliers*. Uno de los más sencillos, que se aprovecha en este algoritmo, suprime para el cálculo todas las muestras cuyo valor se aleje de la media más de dos veces la desviación típica.

Finalmente, se obtiene el ancho de trazo de carácter como la anchura media de las muestras supervivientes. Se ha probado también la posibilidad de asignar el ancho de trazo a la mediana de las muestras en el perfil, pero finalmente se ha optado por la media, porque es más robusta, al utilizar todas las muestras para hacer el cálculo y no sólo una.

3.2.2.3 Métodos descartados

Algunas de las alternativas exploradas con el fin de la obtención del ancho de los caracteres fueron en la línea de la transformada de Fourier, o la transformada discreta del coseno (DCT). En ambos casos, se observó que no había una relación evidente entre la anchura de los caracteres y los máximos u otras propiedades de estas transformadas.

Para ilustrarlo, se muestran a continuación unas imágenes de ejemplo y sus respectivas transformadas de Fourier (figs. 3.2.7 y 3.2.8).

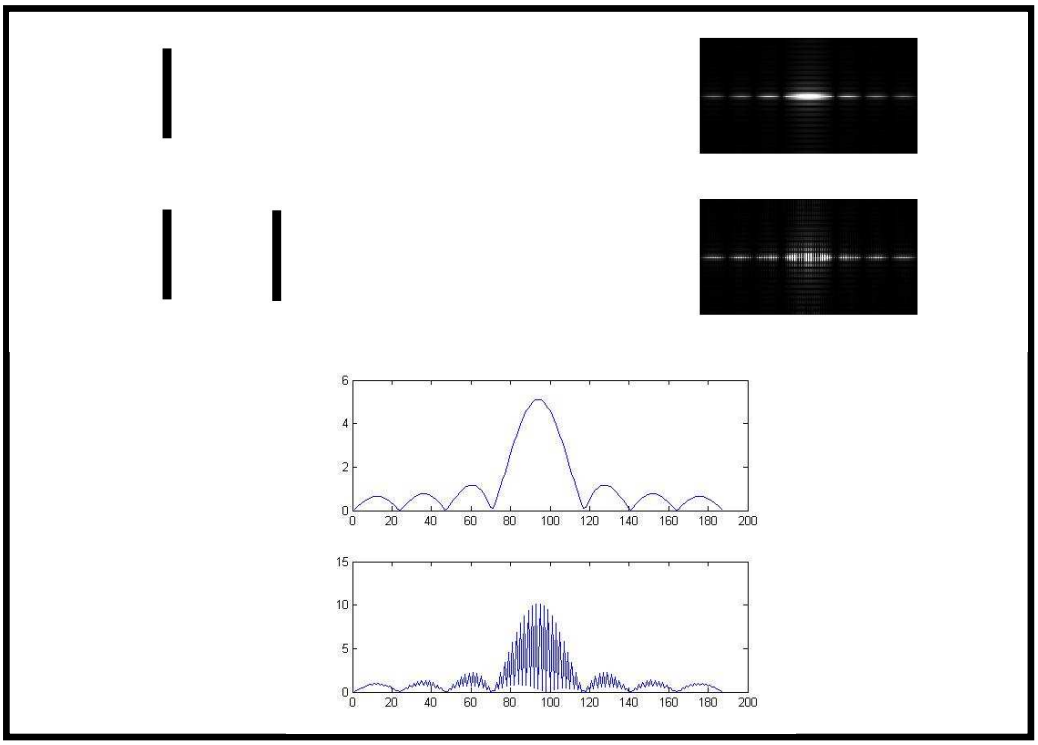


Figura 3.2.7. Imagen y su respectiva transformada de Fourier (arriba) y corte transversal de la transformada (abajo).

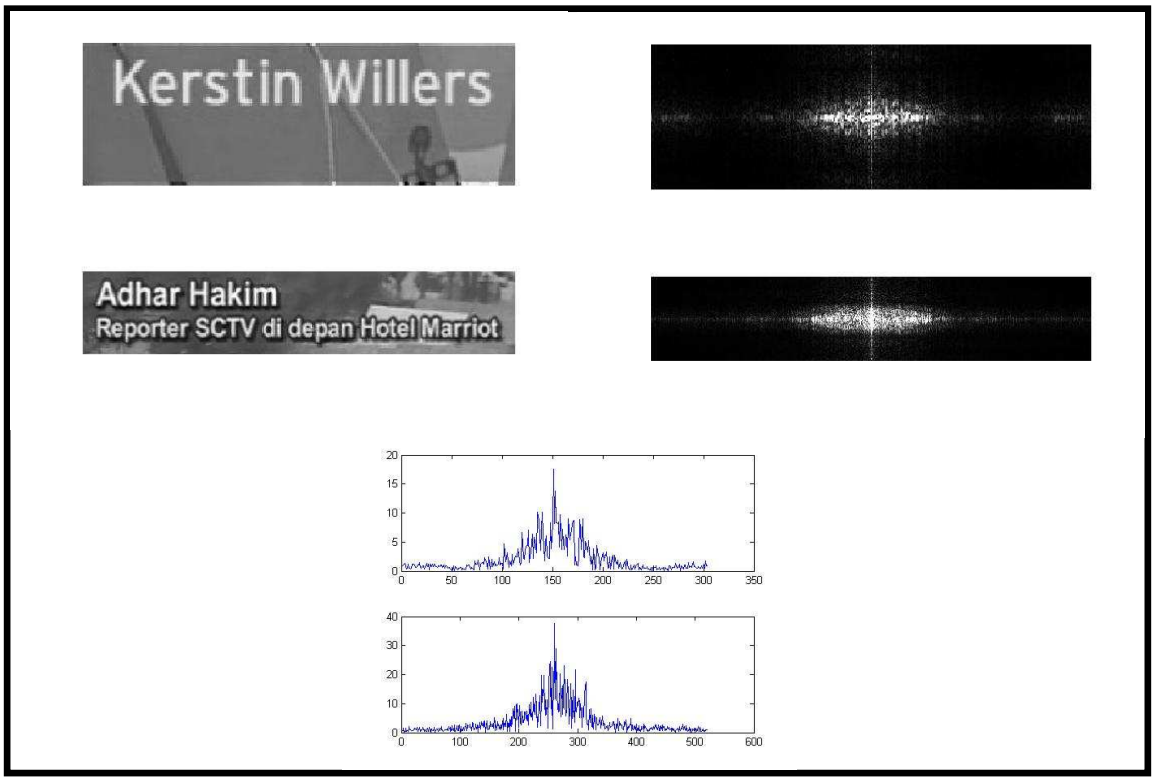


Figura 3.2.8. Imagen y su respectiva transformada de Fourier (arriba) y corte transversal de la transformada (abajo).

3.2.3 Tratamiento morfológico del texto: Top Hat

En esta etapa se efectúan una serie de operaciones morfológicas conocidas como *Top Hat*, que posibilitarán la correcta extracción ulterior del texto y que conforman la esencia del método propuesto. Los pasos anteriores únicamente tienen la finalidad de proporcionar la información necesaria para llevar a cabo esta operación, cuyo resultado debe ser una imagen en la cual el texto haya quedado claramente resaltado con respecto al fondo que lo sustenta.

Operaciones morfológicas sobre imágenes:

La morfología matemática aplicada a imágenes ofrece una visión basada en la forma de las componentes conexas presentes en la imagen, en contraposición con el enfoque tradicional apoyado en los elementos fundamentales o píxeles.

Las operaciones básicas que este tipo de procesamiento de imagen contempla son la dilatación y la erosión. El resultado de aplicar estas operaciones sobre una imagen depende del elemento estructurante utilizado. Un elemento estructurante define qué píxeles del entorno del central se tomarán en consideración para determinar el valor del píxel homólogo en la imagen resultante.

	X	X	X	
	X	C	X	
	X	X	X	

Figura 3.2.9. Ejemplo de elemento estructurante cuadrado de cinco por cinco

Es este un elemento estructurante (fig. 3.2.9), en el que el píxel 'C' es el central, y los sombreados son los que lo conforman. Si efectuamos una dilatación con él, al píxel correspondiente se le asignará el mayor de entre los valores de los píxeles sombreados (el central, por tanto, incluido), en cambio, una erosión le otorgaría el menor. Esta operación se realiza para todos los píxeles de la imagen.

Se denomina dilatación porque da como resultado una imagen en la cual las zonas claras de la imagen (los máximos) se han extendido. Análogamente, la erosión tiene el efecto opuesto, esto es, el crecimiento de las zonas oscuras (los mínimos), o dicho de otro modo, la erosión de las zonas claras.

Los elementos estructurantes con los que se erosiona o dilata pueden tener cualquier forma, pero en este proyecto sólo se emplean en forma romboidal. Éstos parecen ser los que mejor se adaptan a los espacios entre caracteres, aunque no se observa una disminución drástica en la calidad del resultado si se utilizan otras estructuras como círculos o cuadrados.

		X		
	X	X	X	
X	X	C	X	X
	X	X	X	
		X		

Figura 3.2.10. Elemento estructurante de forma romboidal.

La complejidad operacional en tratamiento morfológico se logra con la combinación sucesiva de operaciones básicas, y en este sentido, se consideran habitualmente otras dos operaciones denominadas apertura y cierre. La apertura morfológica se define como la dilatación de la erosión de una imagen con un mismo elemento estructurante, mientras que el cierre es, análogamente, la erosión de una dilatación previa. En imágenes binarias, la apertura elimina pequeños salientes y objetos (máximos) de un tamaño menor que el elemento estructurante, sin modificar apenas la geometría de los objetos que quedan. El cierre elimina agujeros (mínimos) en los objetos.

Si se sigue aumentando la complejidad, se alcanza el siguiente nivel en el que se consideran de nuevo dos operaciones análogas, llamadas *top-hat* y *bottom-hat*. Tal como viene siendo el esquema habitual, el *top-hat* actúa sobre las zonas claras y el *bottom-hat* sobre las oscuras. El *top-hat* es el resultado de restar de la imagen original la apertura de la misma, y el efecto que se obtiene es el resalte de todas las zonas

claras menores que el elemento estructurante. Es esta propiedad la que se explota para resaltar los caracteres presentes en una imagen, ya que éstos se suponen de una anchura de trazo uniforme. Si se ha logrado obtener el ancho de trazo con precisión, es importante generar un elemento estructurante de un tamaño algo mayor que el trazo para asegurar que no "encaja" en el interior de las letras, lo cual desembocaría en errores de binarización, dejando agujeros en las mismas. Hay que tener en cuenta que ciertas letras son más anchas en algunos puntos que el trazo que tienen, como en las juntas o cruces entre trazos. Las pruebas realizadas muestran que un elemento estructurante entre tres y seis píxeles más grande que el ancho del trazo funcionará adecuadamente.

Aplicación del top-hat:

Se muestran a continuación ejemplos del resultado de aplicar estas primeras fases del algoritmo (esto es, estimación del ancho de los caracteres y aplicación de la operación *top-hat* con un elemento estructurante romboidal) sobre imágenes de captions y los histogramas resultantes (ver figuras 3.2.11, 12 y 13). La transformación que se aprecia en los histogramas facilitará la posterior etapa de binarización que se verá en detalle en apartados sucesivos.

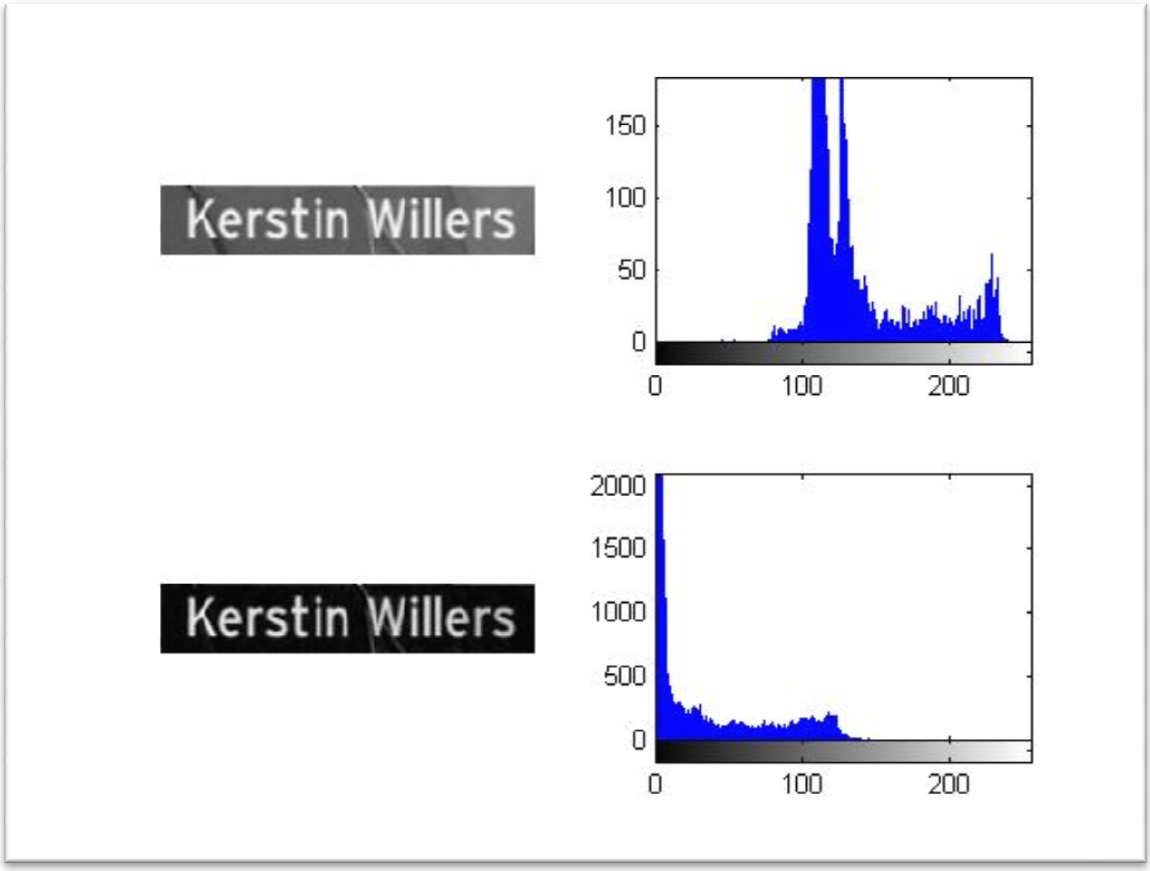


Figura 3.2.11. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas.

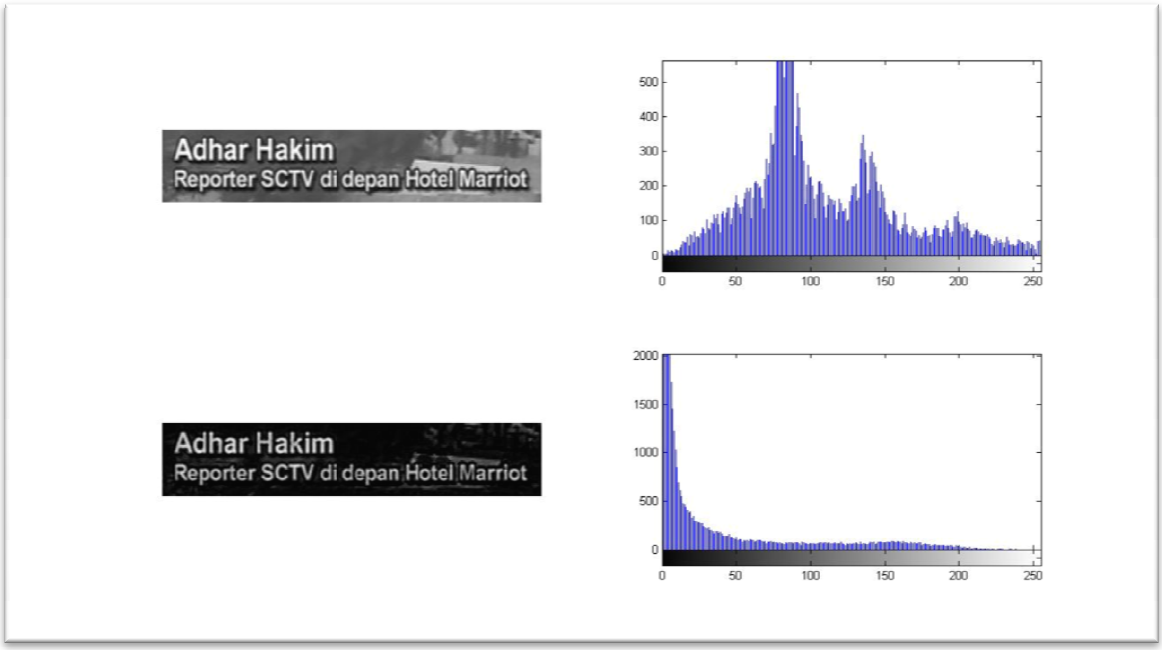


Figura 3.2.12. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas.

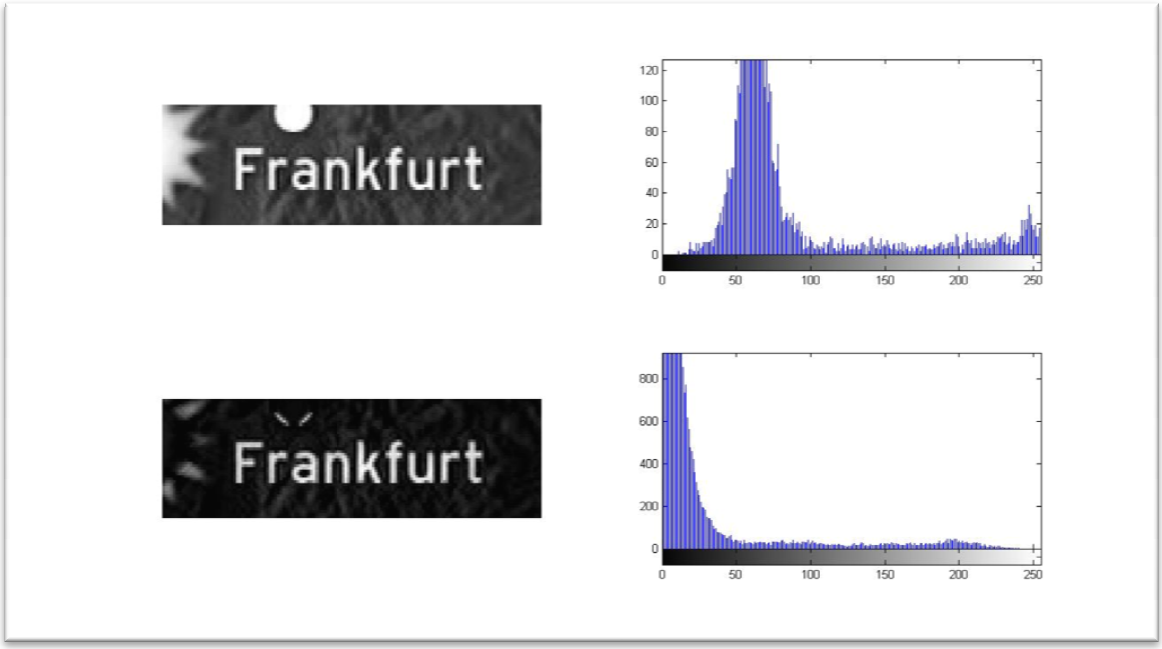


Figura 3.2.13. Ejemplo del resultado de la aplicación de las primeras fases del algoritmo (abajo) sobre una imagen conteniendo un rótulo (arriba) y sus histogramas.

3.2.4 Cambio del tamaño de la imagen

Un software *OCR* convencional (que es el tipo de herramienta considerada en este Proyecto para la etapa final de reconocimiento) está preparado para reconocimiento de caracteres escaneados con una resolución aproximada de 300dpi. En cambio, la resolución comúnmente ofrecida por imágenes o vídeos comprimidos en *MPEG* es mucho menor (obsérvese, por ejemplo, que hasta fechas recientes la resolución estándar para monitores era de 96 dpi). Por tanto, es deseable ofrecer al *OCR* imágenes ampliadas en aras de mejorar su funcionamiento. Existen varias técnicas de interpolación que transforman una imagen en otra mayor, como interpolación por vecino más próximo (*nearest neighbor*), interpolación bilineal, o bicúbica, siendo esta última la que más calidad ofrece y la que, por ello, se ha elegido para este caso.

Todos los métodos de interpolación funcionan de forma similar. Para asignar un valor a un píxel en la imagen destino, se localiza la posición exacta del punto en la imagen original. Después se le atribuye un valor en función del tipo de interpolación, por ejemplo, como media ponderada de los píxeles de alrededor. El tipo de interpolación define qué píxeles se tendrán en cuenta para el cómputo. En el caso *nearest neighbor*, se le asigna al píxel destino el valor del píxel fuente más cercano al punto equivalente. En el caso de interpolación bilineal se usa una vecindad de dos por dos píxeles, mientras que en la bicúbica se usa vecindad de cuatro por cuatro.

Cuanto más compleja es la interpolación mayor es el coste computacional que supone, por lo que es importante buscar un compromiso coste-calidad. En nuestro caso, la importancia de la calidad del resultado es decisiva debido a las exigencias del *OCR* y de ahí la selección de la interpolación bicúbica, que es la más costosa y a la vez la que ofrece la mejor calidad.

Una cuestión a decidir es el momento de realizar esta operación. El tratamiento de imágenes es más costoso cuanto mayores son; por esto es preferible ejecutar la transformación lo más tardíamente posible. En este caso, se lleva a cabo antes de la etapa de binarización, ya que esta es menos efectiva sobre caracteres pequeños. En un carácter reducido, la pérdida de un píxel de borde en la etapa de binarización puede conllevar grandes defectos en su forma, característica observada por los *OCR's* para la determinación del mismo. En cambio, sobre un carácter mayor este efecto es menos apreciable y por tanto es menos probable que produzca una salida errónea.

3.2.5 Binarización

Hasta este momento, lo que se tiene es una imagen ampliada y en la que se han resaltado los caracteres con respecto al fondo. Se trata de una imagen en niveles de gris, de ocho bits por píxel. No obstante, un OCR requiere como entrada una imagen binaria y es determinante para el funcionamiento del reconocedor que la binarización sea buena (es decir, que se marquen claramente los caracteres). Se han desarrollado multitud de técnicas de umbralización automática, y hacer una elección adecuada es muy importante.

Uno de los métodos más conocidos y establecidos es el método de binarización de Otsu ([11]), según el cual se considera que una imagen es una función bidimensional de la intensidad del nivel de gris. Otsu hace uso de técnicas estadísticas para la resolución del problema; en concreto, la **variancia**, que es una medida de dispersión (en este caso referida a los niveles de gris).

Consideramos una imagen cuyos niveles de gris se encuentran entre 0 y G (mayor nivel de gris). Tomamos dos segmentos, el segmento S_1, que comprende los niveles desde 0 hasta t, y el segmento S_2, en el que se incluyen los niveles t+1,…, G. El valor t es el límite entre los dos segmentos, y es el que se debe elegir para obtener una binarización óptima. Para encontrarlo, se calcula la variancia entre los dos segmentos separados por dicho valor, de forma que la variancia entre ambos sea máxima, mientras que la misma medida dentro de los propios segmentos es la mínima, como sigue:

La probabilidad de ocurrencia de cada segmento en la imagen viene dada por:

$$S_1 : P_1(t) = \sum_{g=0}^{t} p(g)$$

$$S_2 : P_2(t) = \sum_{g=t+1}^{G} p(g) = 1 - P_1(t)$$

, siendo p(g) la ocurrencia del nivel de gris g en la imagen (g ϵ [0,…, G]). Como sólo consideramos dos segmentos, la suma de estas probabilidades es uno, ya que cualquier nivel pertenece a uno de los dos. Si denominamos ū al nivel de gris medio de toda la imagen, y $\bar{u}_1$ $\bar{u}_2$ a los niveles medios dentro de los segmentos S_1 y S_2, podemos obtener las variancias dentro de cada segmento como:

$$\sigma_1^2(t) = \sum_{g=0}^{t} (g - \bar{u}_1)^2 \, p(g)$$

$$\sigma_2^2(t) = \sum_{g=t+1}^{G} (g - \bar{u}_2)^2 \, p(g)$$

Si tomamos Q(t) como el cociente entre la variancia entre ambos segmentos ($\sigma_{es}^2(t)$) y la variancia dentro de cada segmento($\sigma_{is}^2(t)$), el valor del umbral óptimo será aquel valor t que maximice dicho cociente:

$$Q(t) = \frac{\sigma_{es}^2(t)}{\sigma_{is}^2(t)}$$

Se calcula la variancia entre segmentos como la suma de las dispersiones de cada segmento con respecto a la media total:

$$\sigma_{es}^2(t) = P_1(t) \cdot (\bar{u}_1 - \bar{u})^2 + P_2(t) \cdot (\bar{u}_2 - \bar{u})^2$$

Por otro lado, la variancia intra-segmento se calcula como la suma de las variancias de cada segmento, por tanto:

$$\sigma_{is}^2(t) = P_1(t) \cdot \sigma_1^2(t) + P_2(t) \cdot \sigma_2^2(t)$$

Y de esta forma se obtiene el umbral deseado que binariza la imagen de forma óptima.

Otra de las opciones disponibles y comúnmente utilizadas es la generación de múltiples hipótesis y trasladar la decisión final al OCR o a algún otro método. Habitualmente, un OCR puede proporcionar el número de caracteres correctamente reconocidos, por lo que es posible seleccionar a posteriori la entrada que obtuvo mejores resultados. Para generar las diferentes posibilidades se usan varios algoritmos cuyo detalle no es objeto de este proyecto, como EM (*Expectation-Maximization*), *Gibbsian EM* o, más comúnmente, *k-means*. Hay una dificultad inherente a este tipo de solución, ya que no todas las imágenes se definen con el mismo número de clases, y unido a esto, requiere un largo tiempo de proceso tanto para la generación de las hipótesis como para la selección de una de ellas. Habitualmente, se necesitan entre cinco y nueve clases para dar con la que representa adecuadamente el texto. Por esto se ha elegido un método de umbral global como el de Otsu.

También existen métodos de umbralización local, que examinan la imagen por partes, decidiendo en cada zona el mejor umbral. En imágenes complejas, pueden resultar costosos ya que requieren efectuar numerosas subdivisiones, y ocasionalmente fallan en descartar letras de zonas en las que no se hallan, ya que no siempre toman en consideración el contraste total de la imagen.

3.2.6 Análisis de componentes conexas

Cualquiera de las técnicas de binarización disponibles introduce eventualmente errores en el resultado. Esto es, no todas las zonas que se obtienen tras la etapa previa son caracteres, sobre todo cuando se tratan captions con fondos complejos tal y como se contemplan en este proyecto. Por este motivo, es necesario ejecutar un análisis de dichas zonas en el que se extraen características de las mismas, tales como altura, anchura, número total de píxeles que contiene, área que comprende, etcétera, con el fin de descartar algunas de ellas que puedan no ser letras.

Existen diversos procedimientos para llevar a cabo este análisis, como algoritmos de crecimiento de zonas o también vasados en el análisis de montículos, como el *watershed*, que trata el valor de los píxeles como un valor de altura, y gestiona la imagen como un conjunto de "valles" y "colinas".

Dichos algoritmos son costosos y contemplan la posibilidad de agregar zonas con niveles parecidos de intensidad y color. En nuestro caso, se parte de una imagen binaria, por lo que un análisis sencillo basta para lograr nuestro objetivo. El algoritmo que se utiliza, basado en el etiquetado de componentes conexas, se describe a continuación.

Se parte, tal como se ha visto, de una imagen binaria en la que sólo hay dos niveles posibles: cero y uno (o cero y doscientos cincuenta y cinco si hablamos de una profundidad de ocho bits) que se corresponden con el negro y el blanco respectivamente. Las zonas en blanco se corresponden con los componentes, y el negro con el fondo. En este punto del algoritmo, la mayoría de los píxeles de la imagen están en negro, y unos pocos en blanco, los cuales equivalen a los caracteres segmentados y a los errores que se desea purgar.

Se recorren todos los píxeles de la imagen, y a cada uno que no es nulo se le asigna una etiqueta. Para determinar dicha etiqueta se comprueba, para cada píxel, si los píxeles adyacentes ya visitados tenían una etiqueta previa. En caso afirmativo se le adjudica la etiqueta de dicho píxel adyacente. Si se determina que hay más de un pixel adyacente con diferentes etiquetas se unifican todas con la menor de ellas, ya que todos estos píxeles pertenecen a la misma componente conexa. Si ninguno de los píxeles adyacentes ya visitados resulta ser no nulo (y poseer por tanto una etiqueta) se le asigna al píxel una etiqueta nueva. Al mismo tiempo que se realiza esta operación, se almacena, para cada componente, la información que permitirá decidir si es un probable carácter o bien un error. Se ha de conocer la altura, la anchura, y el número total de píxeles a los que se asignó una misma etiqueta (esto es, forman parte de la misma componente conexa).

Posteriormente, se requiere que las componentes cumplan una serie de requisitos que son comunes a los caracteres (los umbrales considerados para las restricciones son

producto de un análisis estadístico realizado sobre caracteres de diferentes fuentes comunes, llevados a cabo por [6] y [9]), a saber:

-No estar en contacto con los límites de la *bounding box*: El método de localización de texto delimita una región en la cual el texto se supone centrado. Por esto, cualquier componente que se extienda hasta el límite de la región se supone con alta probabilidad un error. Además, aunque fuera un carácter, si atraviesa dichos límites, tiene un alto riesgo de verse deformado (ya que una o varias partes del mismo se han quedado fuera de la zona de análisis) y por tanto no ser reconocido correctamente por un OCR.

-Contener un mínimo de treinta píxeles: En caracteres de una línea que se han extendido al tamaño visto anteriormente, un punto sobre una 'i' o una 'j' tiene al menos este número de píxeles. Este umbral difiere según la precisión de las *bounding boxes*, el número de posibles líneas de texto por *bounding box* y la posibilidad de que exista más de un tamaño de carácter en el mismo rótulo. En este caso, se ha elegido para escasa restricción y por tanto para abarcar multitud de escenarios.

-Un ratio anchura/altura comprendido entre 0.1 y 2.5: Correspondientes a la 'i' y la 'w', que son los caracteres respectivamente más estrecho y más ancho considerados.

-Anchura total menor que 2.1 por el alto total de la *bounding box*: Cualquier componente que no cumpla esta limitación se considera demasiado ancho para ser un carácter. Esta característica ha sido obtenida del estudio de la 'w' de diferentes fuentes.

-Densidad del componente mayor que 0.2: Esta restricción considera que del rectángulo mínimo que inscribe el componente al menos el veinte por ciento de los píxeles que contiene deben ser no nulos, ya que, como se aprecia en la figura de abajo (figura 3.2.14), la gran mayoría de caracteres se hallan en este rango. En otro caso se descarta.

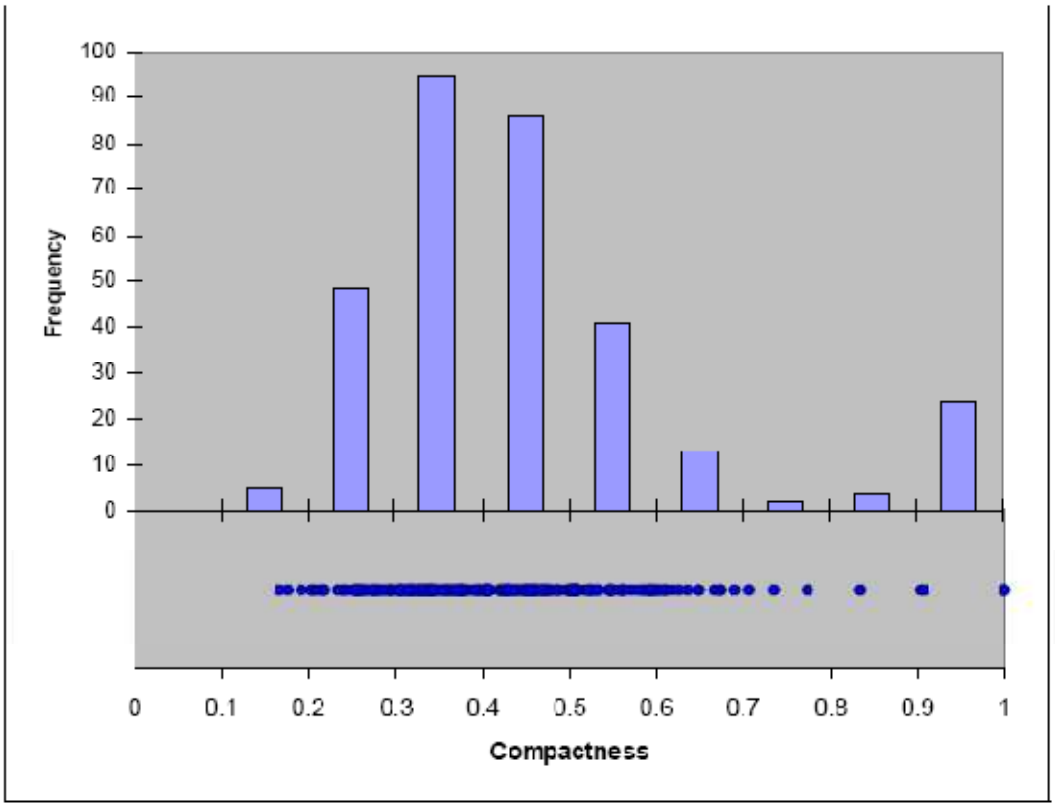


Figura 3.2.14 Distribución de densidad de los 26 caracteres del alfabeto inglés, de fuente arial regular, con diferentes tamaños entre 6 y 22 puntos. Imagen tomada de [9].

Las clases que no superan alguno de estos criterios son considerados errores, y se eliminan de la imagen final.

En rótulos de baja calidad se puede observar cómo este análisis descarta algunos caracteres con errores, por ejemplo, si están en contacto unos con otros o si se han visto alterados por algún añadido no deseable producto de la binarización. Este efecto no se considera perjudicial para el resultado ya que un software *OCR* tendría serios problemas para reconocer caracteres de este tipo.

3.2.7 Reconocimiento de caracteres: OCR

Tal como se viene comentando a lo largo de toda esta etapa de diseño, el objetivo final de las operaciones realizadas es aislar el texto del fondo para su posterior reconocimiento por un *OCR*. Las siglas *OCR* responden a *Optical Character Recognition*, traducible por "reconocimiento óptico de caracteres". Esto es, reconocer un carácter por la forma que tiene.

Los requisitos para la selección de un *OCR* apropiado para este algoritmo incluían, primero el ser un software de uso libre, y segundo poder disponer de librerías para

ejecutarlo sin necesidad de guardar la imagen en disco, operación extremadamente lenta que actuaría en gran detrimento de la eficiencia del programa.

Los *OCR* analizan una figura compuesta por píxeles en blanco y negro y deciden qué carácter (letras, números o símbolos) se corresponde, según un criterio de verosimilitud, con ella. Modernamente se usan algunas otras técnicas que mejoran este rendimiento, pero no se encuentran disponibles en software de uso libre.

Se ha elegido un *OCR* gratuito llamado Tesseract(*http://code.google.com/p /tesseract-ocr/*), hospedado por Google (*www.google.com*), que fue originariamente desarrollado por *Hewlet Packard* entre 1985 y 1995. Desde entonces, ha sufrido numerosas mejoras y ampliaciones, aunque de forma pausada. Entre las ventajas que cuenta, destaca que su fama hace que existan cuantiosas aplicaciones en progreso que deciden incluirlo, y por tanto es relativamente sencillo encontrar cierto soporte técnico no oficial.

Puesto que funciona de forma autónoma e independiente del resto del programa, sólo es necesario alimentarlo con la imagen obtenida después de todo el procesado, y recoger una cadena de texto con los caracteres reconocidos.

4 Diseño software e implementación

En este capítulo se detallan los pormenores de la programación, la división en clases, las funciones de que consta cada una, la operación que cada función realiza y cómo, etc., así como la mención de las variables más importantes y la forma en la que se utilizan.

Debido al marcado carácter de investigación de este proyecto, se requería efectuar gran cantidad de pruebas, por lo que se decidió comenzar a desarrollarlo en un lenguaje flexible como es *MatLab®* (*www.mathworks.com*), para posteriormente, y una vez fijada la estructura, terminarlo en lenguaje C++ introduciendo las correcciones necesarias.

Además, se ha hecho uso de las librerías *OpenCV* (*http://sourceforge.net/projects/opencvlibrary/)*. *OpenCV* es una biblioteca libre de visión artificial originalmente desarrollada por *Intel®* que implementa multitud de operaciones, estructuras de datos, etc., que facilitan el tratamiento de imágenes y vídeos.

4.1 Especificaciones del sistema

El sistema se ha desarrollado en dos versiones distintas. Una de ellas funciona de forma autónoma y que recibe las imágenes fuente con rótulos ya recortados y presentes en un directorio especificado por el usuario. Los formatos que soporta el programa son los mismos que admite la librería *OpenCV*. Estos son: *BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF.*

La segunda se ha sometido a adaptaciones necesarias para incluirla dentro del programa *IVOnLA*, y es éste quien se encarga de localizar, dentro de los vídeos de entrada, los fotogramas que contienen rótulos, y dentro de ellos las *bounding boxes* que los sitúan. A partir de ahí, se suministra esa información al módulo de segmentación y reconocimiento.

4.2 Organización del programa: clases y funciones

El programa consta de dos clases, *csegmentador* y *cmostrar*, ambas declaradas en *csegmentador.h* y *cmostrar.h* respectivamente, e implementadas en *csegmentador.cpp* y *cmostrar.cpp*. La clase *cmostrar* contiene un único método *VerImagen(IplImage*)* que recibe una imagen en formato *OpenCV* y ejecuta una rutina para visualizarla. No es necesario visualizar ninguna imagen durante el proceso ya que es completamente

automático, por lo que esta clase sólo resulta útil para fines de depuración. No obstante, se ha decidido mantenerla en el paquete para facilitar futuras modificaciones.

La clase *csegmentador* ejecuta toda la rutina de segmentación, mediante un método público *segmentar(IplImage*)*.

A continuación se muestra en detalle los contenidos de cada clase:

Clase: *cmostrar*

- **Public**:
  - *VerImagen(IplImage*):* Muestra la imagen seleccionada en una ventana.
- **Private**:
  - No contiene.

Clase: *csegmentador*

- **Public**:
  - *Segmentar(IplImage*):* Ejecuta la rutina de segmentación por etapas.
  - *char *Reconocer(IplImage*,int*):* Devuelve un puntero a la cadena de texto reconocida en la imagen que se le pasa y además actualiza el puntero a entero con el número de caracteres reconocidos.
  - *IplImage* InvertirImagen(IplImage*):* invierte los colores de la imagen fuente y devuelve un puntero a la invertida.
- **Private**:
  - *Struct Tletra*: Almacena información referente a componentes conexas en la imagen: altura, anchura, número de píxeles, etiqueta y posición.
  - *IplImage* InvertirImagen(IplImage* imagen)*: Transforma una imagen en su negativo, es decir, el blanco a negro y los grises a su nivel complementario.
  - *int EstimadorTrazo(IplImage* imagen):* Recibe una imagen y calcula y devuelve el ancho del trazo de los caracteres que contiene.
  - *IplImage* CubicResize(IplImage* imagen,int filas_resize):* Recibe una imagen y el número de filas que debe tener la imagen destino. Devuelve la imagen del nuevo tamaño, manteniendo la relación de aspecto, calculada por interpolación cúbica.
  - *IplImage* ImagenTopHat(IplImage* imagen,int stroke):* Calcula el top hat de un caption cuyos caracteres tienen un trazo de tamaño *stroke*.
  - *IplImage* Umbralizar(IplImage* imagen)*: Umbraliza según el método de Otsu, transformando una imagen en niveles de gris en una imagen binaria.

- *IplImage* CCanalisis(IplImage* imagen)*: Efectúa un análisis de componentes conexas en una imagen binaria y descarta los errores (componentes que no sean consideradas caracteres).
- *uchar GetPixel(IplImage* imagen,int f, int c)*: Devuelve el valor del píxel en la posición (f,c) de la imagen.
- *uchar SetPixel(IplImage* imagen,int f, int c,int value)*: Fija el valor del píxel (f,c) en la imagen al valor value.
- *char *Reconocer(IplImage *imagen,int *num_caracteres_reconocidos)*: Recibe la imagen binaria a reconocer y devuelve una cadena de caracteres con el texto reconocido. Además actualiza el entero *num_caracteres_reconocidos* con la cantidad de caracteres que se reconocieron en la operación. Esta información es útil para saber si un rótulo es válido, esto es, contiene realmente algún carácter.

4.3 Fases del algoritmo

En este apartado se procede a detallar las fases diferenciadas que componen el algoritmo, haciendo especial mención a la información que se requeriría conocer en profundidad en caso de que se decidiese modificar parte del algoritmo.

4.3.1 Lectura de imagen

Las imágenes con las que se alimenta este algoritmo pueden ser suministradas manualmente o bien pueden ser proporcionadas por un programa de localización de fotogramas con texto en vídeo, que proporciona el fotograma y las coordenadas de la(s) *bounding box(es)* dentro del mismo. Sólo se consideran imágenes en escala de grises, ya que se ha decidido que las operaciones morfológicas sobre las que se fundamenta este proyecto se apliquen sólo a la banda de luminancia. Por tanto, todas las imágenes se transforman a escala de grises después de su lectura en caso de ser en color.

4.3.2 Decisión texto normal o inverso

El uso de operadores morfológicos que este algoritmo implementa obliga a tomar decisiones distintas si el texto es lo que se considera normal (caracteres claros sobre un fondo oscuro) o inverso (al revés). Los métodos analizados para discernir este aspecto no han podido satisfacer las condiciones de efectividad mínimas, debido a la gran diversidad de tipos de letra y fondos sobre los que se ha testado el algoritmo. Por tanto, ante los problemas encontrados a la hora de dar con un criterio inicial suficientemente preciso y robusto para determinar si un rótulo contiene texto normal

o inverso, se ha decidido (a pesar de incremento en coste computacional que supone) apoyarse en el reconocedor de caracteres y, por lo tanto, comprobar los dos supuestos para asegurar el mejor resultado posible.

Tras la lectura de la imagen, se genera una imagen de igual tamaño, y que se rellena con el inverso de la imagen original, esto es, el negro pasa a blanco y viceversa, y los niveles intermedios de gris se van a su complementario. Se ejecuta el algoritmo de segmentación y reconocimiento sobre ambas imágenes, y se hace recuento de los caracteres válidos reconocidos en cada una de ellas. Se consideran caracteres válidos a los números 0…9, letras A…Z y a…z. Al final, sólo se considera la salida que ha generado mayor número de caracteres válidos, descartándose la otra como no válida.

4.3.3 Estimación del ancho del trazo de los caracteres

Esta fase del algoritmo se divide en dos partes, ambas implementadas en la función *EstimadorTrazo*, que recibe un dato del tipo *IplImage** de *OpenCV*. En la primera parte se ha de estimar las filas en las que empieza y termina una línea de texto. (Nota: puesto que se utiliza tratamiento morfológico, no se garantiza el correcto funcionamiento del algoritmo en *captions* que contengan más de una línea de texto en la que los caracteres muestren tamaños diferentes. Aunque esto ha probado no ser un obstáculo en numerosas ocasiones, el algoritmo está diseñado para el uso de *captions* con el mismo tipo de texto en todas sus líneas). En la segunda parte se extrae el ancho de trazo.

En ambos casos, hay que calcular el perfil de proyección de la zona bajo examen, por lo que se declaran dos variables (*arrays*) *perfil_horizontal* y *perfil_vertical*, del tamaño de las filas y las columnas de la imagen respectivamente, que los almacenarán.

El perfil de proyección de una imagen en una de sus direcciones principales es la suma del valor de intensidad de todos los píxeles alineados en esa dirección. Tiene, por tanto, el mismo número de posiciones que filas o columnas (según sea horizontal o vertical) tiene la imagen, y cada una de estas posiciones contiene la suma de la intensidad de todos los píxeles en su fila/columna correspondiente.

Inicialmente, se procede a calcular el perfil de proyección horizontal de la imagen, que facilitará información acerca de la posición de las líneas de texto dentro de la *bounding box*. Esta operación se lleva a cabo sobre una imagen resultado de pasar un filtro *Canny* a la imagen original. La imagen filtrada mostrará líneas de un píxel de ancho donde haya un borde en la imagen original, lo que ocasiona que sea más sencillo localizar las filas con más bordes (supuestas pertenecientes a líneas de texto) que en la imagen fuente. Para obtenerla se usa la función *cvCanny* de *OpenCV*.

La función *cvCanny* recibe la imagen fuente, la imagen destino, el umbral mayor, el umbral menor y el tamaño del *kernel* del filtro *sobel* que esta función implementa internamente. La imagen destino se crea previamente con *cvCreateImage*, del mismo tamaño que la original, con 8 bits de profundidad, y se llama *imCanny*. En cuanto a los umbrales, se ha elegido el umbral mayor de un valor de doscientos cincuenta y cinco y el menor de diez. Estos umbrales son dependientes de la aplicación y tienen que ver con el contraste que tiene que tener un borde para ser localizado y marcado. En este caso, el texto en un rótulo suele tener gran diferenciación con el fondo y por tanto es muy probable que se marque en la mayoría de los casos. Esto significa que no hay gran dependencia de los umbrales en esta aplicación y que, por tanto, casi cualesquiera valores en un rango razonable deberían servir. El tamaño del *kernel* tiene que ser uno, tres, cinco o siete, y se le ha otorgado el valor intermedio de tres, ya que por las mismas razones que las antes expuestas, el algoritmo no es demasiado sensible a la precisión de estos umbrales.

Una vez hecho esto se rellena cada posición de la variable *perfil_horizontal* con la suma de píxeles en blanco de la fila correspondiente, teniendo en cuenta que un píxel en blanco vale doscientos cincuenta y cinco y que se normaliza ese valor a 1.

A continuación se calcula la media de la suma de los bordes de todas las filas para, comparando con este valor, saber si una fila tiene alta densidad de bordes o no. Para ello, se recorren las posiciones del perfil de proyección hasta dar con dos posiciones contiguas cuya densidad de bordes supere la media. Se buscan dos porque un borde aislado y horizontal en la imagen original no perteneciente a caracteres, hecho bastante frecuente, presentará un perfil muy alto, pero como los bordes tienen ancho de un píxel no es probable que la línea siguiente a su vez muestre un perfil elevado (ver fig. 4.3.1 y 4.3.2, en la que la línea de texto empieza en torno a la fila 60, y el error en torno a la fila 20 no se toma como inicio de línea porque no hay dos posiciones consecutivas con valores superiores a la media, en rojo). Cuando se han localizado estas dos posiciones, se recorre hacia atrás buscando una en la que el perfil caiga por debajo de la mitad de la media, momento en que se habrá salido de la línea de texto. Se usa un *flag* '*init*' para que sólo se realice esta operación una vez. Después se continúa recorriendo las filas hasta que se cumple el mismo criterio para posiciones posteriores a las dos que son mayores que la media, momento en el que se ha alcanzado la posición correspondiente a la fila final de la línea de texto. Un criterio adicional obliga a que la separación entre las filas inicial y final sea mayor que diez. En caso contrario se continúa buscando. Esta limitación asegura que se seleccionan líneas cuyos caracteres miden al menos 10 píxeles de alto, lo cual es adecuado para garantizar la legibilidad del texto.

Figura 4.3.1 Rótulo filtrado con un filtro de bordes *Canny*, y con errores en el fondo.

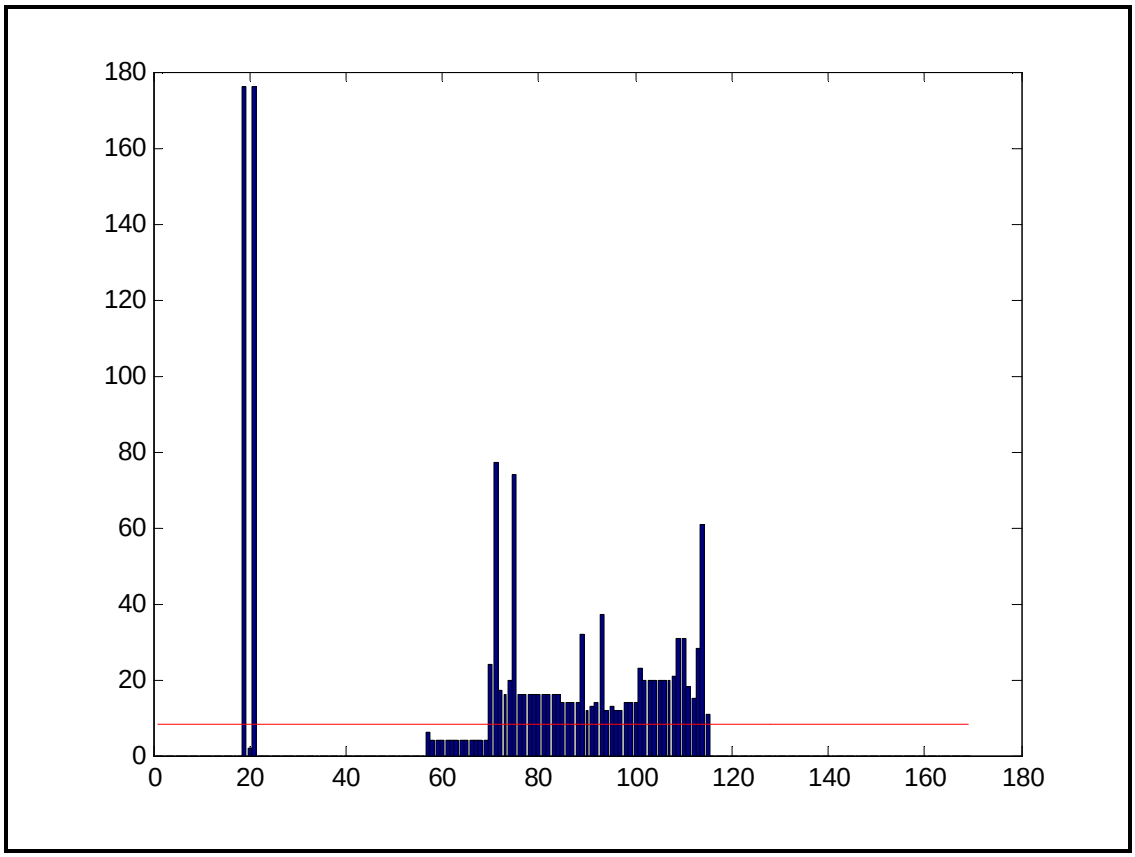


Figura 4.3.2 Perfil de proyección horizontal. En rojo, el nivel medio.

Normalmente, no se ven incluidos entre estas líneas los caracteres completos (por ejemplo las zonas altas de las 'd', las 'h', etc.). La ocurrencia de este suceso depende del número de letras "altas" que haya en la fila. Si son escasas, derivará en que las partes altas de las letras no se utilizarán para la generación del perfil vertical. Esto no supone un problema para el algoritmo, bastando con que la mayoría del área de los caracteres esté contenida. Con esto concluye la primera parte.

Se continúa rellenando el perfil vertical, que permitirá discernir el ancho de trazo propiamente dicho. Hay que actuar en adelante sobre la imagen original, descartando la imagen de contornos que se había considerado en la primera parte, ya que en este caso nos interesa que todos los píxeles aporten su valor al perfil. Tal como se hizo previamente, se rellena en *perfil_vertical* el valor acumulado de todos los píxeles pertenecientes a cada columna, obteniendo así un perfil de picos en las zonas donde hay letra y valles en los espacios entre ellas.

Para poder aislar el ancho de estos picos (que es también el ancho del trazo) hay que contar el número de posiciones que comprenden. Se facilita esta tarea restando la media de todas las posiciones, con lo que se logrará que entre pico y pico haya ceros en vez de valores pequeños. Si al restar la media el resultado es menor que cero hay que truncarlo, y de esta forma, se puede recorrer el *array* contando el número de posiciones no nulas entre conjuntos de valores nulos. Estos valores se almacenan en la variable *t_trazo[].*

Se dispone hasta ahora, según lo visto, de las anchuras de los trazos de las letras según el perfil de proyección vertical. Como se vio en el Capítulo 3, la simple media de estos valores no representa con precisión el ancho deseado, por lo que se implementa una media robusta descartando algunos valores para su cálculo. Para esto se calcula la desviación típica de los valores de anchura encontrados, y se descarta todo aquél que se aleje de la media más de dos veces la desviación típica y se calcula la media de los restantes. En la variable '*stroke*' se almacena el valor del trazo calculado.

Como paso final se libera la imagen '*imCanny*' y se devuelve el valor calculado.

4.3.4 Top-Hat de la imagen

En esta etapa se calcula el *Top-Hat* de la imagen, en una función que se llama *ImagenTopHat*. Recibe la imagen sobre la que se va a actuar, en formato de imagen de *OpenCV*, y el tamaño de trazo extraído en la etapa previa.

Inicialmente, hay que generar un elemento estructurante adecuado. Como se vio en el Capítulo tres, se usan elementos de forma romboidal. *OpenCV* dispone de una función que compone dichos elementos estructurantes, pero no contempla entre sus posibilidades automáticas esta morfología, debido a lo cual hay que generarlo específicamente. La función que lo hace devuelve una estructura de tipo *IplConvKernel*, y se llama *cvCreateStructuringElementEx*. Recibe el alto y ancho del elemento estructurante, los *offsets* horizontal y vertical que presenta el píxel central, la forma del elemento y un *array* de enteros. En caso de que dicha forma se genere prescindiendo de los literales que le otorgan una prefijada, hay que almacenar en el *array* los valores no nulos del elemento por filas. Como se vio anteriormente, el elemento estructurante tiene que ser ligeramente mayor que el trazo para que no

encaje en el interior de las letras. También se adecúa el alto y el ancho del elemento a un valor impar, para mantener la simetría respecto al píxel central. Juntando estas dos condiciones, se suma al tamaño del trazo cinco píxeles si es par y seis si es impar, y éste será el ancho y alto máximo del elemento. El número de píxeles que se ha de añadir al tamaño del trazo para que sea suficientemente grande como para no encajar dentro de las letras es variable y depende del tipo de letra del rótulo. Según las pruebas realizadas, cinco píxeles de diferencia es suficiente para garantizar la integridad del resultado en la inmensa mayoría de los casos.

El *array* de enteros donde se guardan los valores se ha denominado '*shape'*, y hay que rellenarlo apropiadamente con objeto de que el elemento resultante sea efectivamente romboidal. Un elemento estructurante romboidal de tamaño máximo impar es de la forma (ejemplo cinco por cinco (Fig. 4.3.3)):

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

Figura 4.3.3 Elemento estructurante romboidal de cinco por cinco píxeles

Por tanto, el *array 'shape'* debería contener 0010000111011111010111000100. Se ha planteado este problema en forma de ejes cartesianos. Si tomamos el píxel central como origen de coordenadas, y los desplazamientos horizontal y vertical como X e Y, se cumple que para un rombo, el valor absoluto de X mas el valor absoluto de Y tiene que ser igual o menor que dos. Generalizando este razonamiento para cualquier tamaño se consigue dar los valores adecuados a la variable '*shape'*. Tras esto se llama a la función que genera el elemento estructurante y se guarda el resultado en la variable '*diamond*'.

El siguiente paso es efectuar la operación *Top-Hat* propiamente dicha. Se crea, para ello, una imagen denominada '*imTopHat'*, del mismo tamaño que la original. Afortunadamente, *OpenCV* implementa esta operación en una función llamada *cvMorphologyEx*. Recibe las imágenes fuente y destino, el elemento estructurante, el tipo de operación (*Top-Hat*) y el número de iteraciones, en este caso una. Consecutivamente se devuelve la imagen resultado, concluyendo así el cálculo del *Top-Hat* de la imagen.

4.3.5 Aumento de la resolución de la imagen

El objetivo de esta fase del algoritmo es aumentar la resolución de la imagen para mejorar el resultado de la etapa de binarización. De nuevo, *OpenCV* permite llevar a cabo esta operación de forma sencilla mediante una función interna, llamada *cvResize*,

a la que se debe pasar las imágenes fuente y destino y el tipo de interpolación. La función adapta la imagen fuente al tamaño de la imagen destino. Sólo se agranda la imagen en caso de que originariamente posea menos de cien filas de alto. Ver el Capítulo tres para más detalles acerca de la conveniencia de realizar esta operación.

4.3.6 Umbralización

Esta etapa se implementa en la función *Umbralizar(IplImage* image)*, que recibe la imagen sobre la que se ha de actuar en formato imagen de *OpenCV,* y su finalidad es eliminar de la imagen todos los componentes pertenecientes al fondo, dejando únicamente los caracteres.

El método de umbralización elegido es el llamado método de Otsu. Es un método de umbral global que divide los píxeles en dos clases y maximiza la varianza entre ellas (para más detalles ver el Capítulo tres). De nuevo, *OpenCV* proporciona una función que implementa esta operación. Se genera la imagen destino (del mismo tamaño que la original y 8 bits de profundidad) y se denomina '*imBinarizada*'. La función que binariza se llama *cvThreshold*, y requiere las imágenes fuente y destino, dos valores fijos y un literal indicando el tipo de umbralización. El primer valor representa al que irán a parar los píxeles menores que el umbral, y el segundo es el valor por el que se sustituyen los píxeles mayores. En este caso, el primero es cero y el segundo doscientos cincuenta y cinco (negro y blanco, respectivamente). El literal que indica a *cvThreshold* que debe implementar el método de Otsu es "CV_THRESH_BINARY |CV_THRESH_OTSU" (véase ayuda de *OpenCV* para más información).

Entonces, *imBinarizada* tendrá algunos píxeles a cero y otros a doscientos cincuenta y cinco según los umbrales antes definidos, es decir, es una imagen binaria cuyas zonas blancas se corresponden supuestamente con los caracteres. No obstante, siempre se introducen errores en la binarización debidos a elementos del fondo que por sus características morfológicas son seleccionados por el algoritmo de umbralización. Estos errores entorpecerían la función del *OCR*, dificultando o imposibilitando el reconocimiento, lo que nos lleva a la etapa siguiente: el análisis de componentes conexas.

4.3.7 Análisis de componentes conexas.

Se parte de una imagen binaria resultado de la umbralización antes vista. En esta etapa se lleva a cabo un análisis sencillo de componentes conexas, de las cuales se descartarán aquellas que no satisfagan los requisitos necesarios para ser consideradas caracteres.

La finalidad es otorgar a cada píxel de la imagen original una etiqueta, de forma que todos los píxeles que pertenecen a un mismo componente compartan una etiqueta. Una vez que se han etiquetado, se puede extraer de ellos la información que servirá para discernir si se consideran o no caracteres, y eliminarlos en caso necesario.

Se crea una imagen destino de un canal, ocho bits de profundidad, e igual tamaño que la original con la función *cvCreateImage* de *OpenCV*, llamada '*imConnected'*. Esta función no inicializa el valor de los píxeles a cero, por lo que es necesario hacerlo expresamente, para mantener la uniformidad de los píxeles no etiquetados. En esta imagen se escriben las etiquetas de cada píxel, y es la única que se modifica.

Los elementos que se necesitarán son un *array* de estructuras asociadas a cada componente donde se almacenará la información concerniente a dichos componentes, llamado '*letras'*, y una tabla de equivalencia entre clases que define para cada etiqueta su clase equivalente menor (variable '*tabla'*).

Se recorre la imagen por filas, estudiando el valor de cada píxel en la imagen original. Los píxeles nulos se dejan como están, puesto que no forman parte de ningún componente. Para cada píxel no nulo, se observan los valores de los píxeles adyacentes ya visitados, esto es, en una vecindad de 8 píxeles, los tres colindantes en la fila de arriba y además el de la izquierda de la misma fila. Se guardan sus etiquetas en las variables '*lu*', '*u*', '*ru*', y '*l*', que son acrónimo de *left-up, up*, etc. Si alguno de éstos es no nulo, tiene que tener un valor asociado en la imagen de clases (puesto que ya ha sido visitado), y entonces se le asigna al píxel actual la misma etiqueta (ya que pertenecen al mismo componente). En caso de haber varios, se le asigna la **menor** etiqueta de entre las posibles, y se actualiza la tabla de equivalencia. ¿Por qué actualizar la tabla? Porque en cada momento sólo se modifica en la imagen de clases el valor del píxel bajo examen, ya que si se altera la etiqueta de uno de los colindantes se podría perder el nexo con otros que sean a su vez adyacentes a él. Por esto hay que variar la tabla, en la que se estipula que todos los píxeles con la etiqueta 'E' pertenecen a la clase equivalente definida por *tabla[E]*. Posteriormente se sustituirán todas las etiquetas por la menor de sus equivalentes para que todos los píxeles de la misma clase tengan la misma etiqueta.

En caso de no haber ningún píxel colindante ya visitado, se le otorga al píxel actual una etiqueta nueva, y se continúa el proceso hasta el final.

A continuación se procede a reemplazar todos los píxeles etiquetados por su etiqueta equivalente menor, guardada en *tabla[]*, y además se rellena la información que se precisa sobre las clases. Para esto hay que recorrer toda la imagen de nuevo, haciendo la sustitución oportuna (esto es, si el pixel tiene etiqueta 'E', se reemplaza por *tabla[E]*). Se ha creado un *array* de estructuras llamado letras, cuyos miembros contienen los campos *'alto', 'ancho', 'xmin', 'xmax', 'ymin', 'ymax', 'label'* y *'nump'*.

Para cada píxel (después de modificarlo si era preciso), se comprueba si existe un miembro de la estructura '*letra*' que posea su misma etiqueta. En caso negativo se inicializa un miembro de la estructura, en el que *letra.xmin* y *letra*.*xmax* se inicializan a la columna actual, *letra*.*ymin* y *letra.ymax* a la fila actual y *letra.nump* a uno. Si hubiera un miembro cuya etiqueta coincide con la del píxel, se modifica la información asociada (esto es, actualizar *letra.xmin*, *letra.ymin*, etc. en caso de que sea necesario, y se suma uno a *letra.nump*). Hemos logrado así tener una imagen en la que todos los píxeles en una componente conexa tienen la misma etiqueta y un *array* con una estructura por cada etiqueta en la que se guarda la información de tamaño y posición de cada una de esas componentes conexas.

Se dispone ahora de las características de cada componente conexa, a falta de determinar, para cada miembro de la estructura, *letra.alto* como la resta entre *letra.ymax* y *letra.ymin*, y se procede de la misma manera con el campo *letra.ancho*. En el mismo bucle en que se realiza esta operación se determina qué componentes se borrarán. Para cada miembro (componente), se define una variable '*ratio*' en la que se almacena la relación entre el ancho y el alto de cada carácter. Posteriormente se comprueba que:

a) El carácter no toca los bordes de la imagen: Es decir, *letra.xmin* o *letra.ymin* no son cero, ni *letra.xmax* es igual al número de filas ni *letra.ymax* es el número de columnas.
b) El ratio ancho/alto está entre 0.1 y 2.5: Se ha de comprobar que el alto del carácter no es cero, en cuyo caso se le asigna a '*rate*' el valor cero y por tanto se elimina.
c) El ancho del carácter no es superior a 2.1 por el número de filas.
d) El número de píxeles no es inferior a 30.
e) La densidad del carácter es mayor que 0.2: Calculada como el número de píxeles por unidad de área.

Las justificaciones de estas condiciones se han explicado en el apartado 3.2.6.

En caso de que alguno de estos requerimientos se incumpla, la etiqueta de la clase en cuestión se añade al *array clases_a_borrar* y se aumenta la variable *nclases_a_borrar*, que lleva la cuenta del número de clases que se descartan.

Concluido este paso, sólo resta eliminar de la imagen de clases los componentes descartados. Necesitamos por tanto recorrer por última vez toda la imagen de clases. Para cada píxel, se compara su etiqueta con la lista de etiquetas a borrar y, en caso de encontrarse, se elimina de la imagen (Ver figuras 4.3.3 y 4.3.4).

Figura 4.3.4 Imagen umbralizada antes del análisis de componentes conexas.

Figura 4.3.5 Imagen umbralizada después del filtrado de componentes y binarizado final con caracteres negros y fondo blanco.

Como última fase, hay que binarizar la imagen de clases. Hasta este momento, las etiquetas (los valores de los píxeles en la imagen de clases) se comprendían entre uno y el número total de clases. Se requiere que todos los píxeles no nulos pasen a tener valor doscientos cincuenta y cinco (blanco en una imagen de un canal con ocho bits de profundidad). Para esto se hace uso de la función *cvThres* de *OpenCV*, con umbral 1 y valor final 255, pero en este caso se usa el literal *CV_THRESH_BINARY_INV*, lo que invierte el tipo de binarización, para que los caracteres queden finalmente de color negro y el fondo blanco. Esta es la única forma en que se puede alimentar al *OCR (Figura 4.3.4)*.

4.3.8 Reconocimiento de caracteres.

En esta etapa se realizan los últimos ajustes y se proporciona la imagen resultado de las etapas anteriores a un programa de reconocimiento de caracteres (*OCR*) que traduce la imagen a texto editable. Todas las etapas previas están encaminadas a tratar la imagen para facilitar el trabajo al *OCR* y maximizar la cantidad de caracteres correctamente reconocidos. Se implementa en la función *Reconocer,* de la clase *segmentador*.

El *OCR* seleccionado para la tarea se llama *Tesseract (V 2.03)*. Este software es *OpenSource*, y podrá ser sustituido en el futuro por versiones sucesivas. Se puede descargar gratuitamente de *Google code* (http://code.google.com/p/tesseract-ocr/). Viene en forma de proyecto de *Visual C++*, y ha compilado correctamente en la versión 8.0. Una vez creado el proyecto y compilado, *Tesseract* ofrece una librería para llamar a sus funciones desde un código externo, que se encuentra en *Tessdll.dll*. Hay que dar,

por tanto, al programa acceso al directorio donde se encuentre dicho archivo, o copiarlo al directorio principal del mismo. De igual manera, se debe incluir en el proyecto el archivo *tessdll.h*, y los ficheros de cabecera necesarios que cuelgan de éste, a saber: *ocrclass.h, platform.h, host.h, applybox.h, mfcpch.h*. Asimismo, se debe agregar al proyecto la librería *Tessdll.lib*, en las propiedades del proyecto, como directorio adicional de librerías (pestaña *linker/general*) y en las dependencias adicionales (pestaña *linker/input*).

Además, se deben descargar en un paquete separado los archivos correspondientes al idioma que se desea reconocer. Hay varios disponibles, y debe existir al menos uno, o de lo contrario el programa se finaliza automáticamente. Deben colgar del directorio principal del proyecto (carpeta *debug* o *release*, donde se genera el ejecutable "*captions.exe*"), dentro de una carpeta que se llame *tessdata*.

NOTA: Se ha detectado un *bug* en esta versión de *Tesseract* que detona un *assertion failure* en el archivo *isctype.c* al alimentarlo con ciertas imágenes. Para corregirlo, se ha de sustituir, en el archivo *dawg.cpp* (que pertenece al *Tesseract*) la línea 148*: isalpha (dummy_word [char_index])* por *isalpha ((unsigned char)dummy_word [char_index]).*

Se requiere llevar a cabo otro pre procesado antes de llamar al *OCR*, esta vez referente a la imagen. La razón de incluir las librerías *OCR* en el código del algoritmo es la eficiencia temporal, ya que de esta forma no es necesario escribir la imagen a disco para que sea inmediatamente vuelta a cargar por el *OCR* para su proceso. Se puede alimentar al *OCR* con una cadena de bytes en "bruto", especificando el tamaño de filas y columnas. He aquí que *OpenCV*, formato utilizado a lo largo de todo el algoritmo, no almacena la imagen como una cadena de *bytes*, sino que introduce después de cada fila una serie de *bytes* para tratamiento interno. Es preciso separar esos bytes y descartarlos. En la variable '*datos'* se introduce dicha información, que ya se encuentra preparada para alimentar al *OCR*.

Se procede entonces a efectuar el reconocimiento. Para ello se requiere crear un objeto de la clase *TessDllAPI* al que se le pasa el código del idioma de reconocimiento, 'eng' para inglés, 'spa' para español, etc. No se puede, por tanto, crear automáticamente según el idioma del *rótulo*. El inglés ha demostrado ser igual de útil para su propia lengua que para las demás, por tanto es recomendado cargar siempre éste. A continuación se llama al método *BeginPageUprigth*, que recibe el ancho y alto de la imagen, la cadena de datos brutos y la profundidad (8 bpp en este caso):

*TessDllAPI::BeginPageUprigth(int width, int height, uchar *data, int depth);*

Seguidamente, se llama al método *Recognize_all_Words(void)*, que efectúa el reconocimiento y devuelve un puntero a una estructura de tipo *ETEXT_DESC* que contiene la salida. El campo '*count'* de la estructura contiene el número de caracteres, y el campo *text* es un *array* de estructuras de tipo *EANYCODE_CHAR* con los caracteres

leídos, cuyo subcampo *char_code* contiene el código *ASCII* del carácter. No se tienen en cuenta en esta estructura los espacios en blanco. Si se quieren representar, otro subcampo de la estructura llamado '*blank*' contiene el número de espacios vacíos delante de cada carácter. Se tienen, pues, todos los caracteres reconocidos, que se almacenan en un *array* llamado '*buffer*', que contiene el texto reconocido del rótulo, y por tanto, el fruto de todo el proceso.

También se puede llamar a las funciones sin crear previamente un objeto. Para ello se dispone de un *wrapper* (envoltorio), generado en las funciones *TessDllBeginPageUprightBPP*, al que se deben pasar los mismos parámetros vistos anteriormente más el idioma, ya que al no crearse un objeto, no se dispone de él, y *TessDllRecognizeAllWords*, que funciona igual que la anteriormente vista.

4.4 Integración

Con objeto de aprovechar la funcionalidad del módulo de localización de fotogramas con texto en dominio comprimido y *bounding boxes*, se ha integrado este algoritmo como módulo en otro programa de análisis de vídeo denominado *IVOnLA*.

IVOnLA es un sistema que extrae información de vídeos en dominio comprimido y en tiempo real. La salida que se obtiene es una descripción del contenido del vídeo, que incluye su estructura temporal, descriptores de movimiento global o de cámara, máscaras de segmentación basadas en movimiento, imágenes DC y coeficientes DCT, y al que se pretende añadir, con este módulo, la capacidad de extracción del texto que se muestre en rótulos superpuestos. Uno de los módulos de que ya constaba previamente *IVOnLA*, se encarga de localizar fotogramas en vídeo comprimido que reúnen las características necesarias para contener un rótulo, y además proporcionar unas coordenadas que delimitan el rótulo (*bouding boxes*). Estas "cajas" vienen definidas por el módulo de localización como objetos de un tipo denominado *CaptionBox*, que contiene sus coordenadas en bloques (y no en píxeles) y un campo *bActive* que determina si es válida. En caso de estar *bActive* desactivado, significa que la *CaptionBox* se localizó pero fue posteriormente descartada en una etapa de refinamiento, por lo que se debe ignorar para el reconocimiento de texto.

Para poder incorporar este programa a la estructura principal del *IVOnLA* ha sido necesario introducir ligeras modificaciones.

Se ha incluido en la estructura del *IVOnLA* una clase llamada *TextExtractor*, y un objeto de dicha clase denominado *CaptionTextExtractor*. Mediante el método *ExtractCaptionTexts*, se llama al módulo de segmentación y reconocimiento. El algoritmo requiere que se le proporcione el fotograma, las coordenadas de la

bounding box (que contienen presumiblemente el texto) y una estructura en la que se rellena el texto que se ha reconocido.

Las *bounding boxes* se presentan con formato de los bloques en los que empiezan y los que terminan, pero la única imagen suministrada es el fotograma completo. Por tanto, se activa en la imagen leída con *OpenCV* un campo de la estructura *IplImage* (que contiene la imagen) que se denomina *ROI (Region of Interest)* con las coordenadas de la *bounding box* dentro del fotograma. Mediante esta activación, se logra que las operaciones con la imagen actúen únicamente sobre la parte que se encuentra dentro de dicha *ROI*. Por ejemplo, la función *CvCopyImage* copiará en una imagen destino sólo la región seleccionada de la imagen fuente. Esta es la técnica que se utiliza para obtener una imagen completa recortada del fotograma según los límites de la *bounding box*. Se crea, pues, una imagen nueva del tamaño definido por la *bounding box* (Nota: al tomar las coordenadas de las *bounding boxes*, se agranda 4 píxeles por cada lado para intentar asegurar que los caracteres no se ven seccionados por los límites de la misma) con el comando *CvCreateImage*, y posteriormente se copia, con la función antes mencionada, desde la imagen original con la *ROI* activa hacia ésta.

Se ha generado un objeto de la clase *segmentador*, que implementa todo el algoritmo visto, y mediante a una llamada a su método *segmentar(IplImage *imagen, char *texto)* que recibe la imagen a segmentar y la cadena de texto que se actualizará con el resultado, se comienza la ejecución del algoritmo.

Sólo se ejecuta el algoritmo para las *CaptionBoxes* (nombre de los objetos que representan las *bounding boxes*) que son válidas, es decir, tienen su campo *bActive* activado. En caso contrario se continúa examinando *CaptionBoxes* hasta que se terminen. Un fotograma puede tener varias *CaptionBoxes*, que pueden o no ser válidas.

No todas las *CaptionBoxes* tienen que generar una salida. Para cada *CaptionBox*, se ejecuta el algoritmo sobre la imagen original y la inversa. Si ambas generan salida (número de caracteres reconocidos mayor que cero) se devuelve la salida con mayor número de caracteres reconocidos. Si una genera salida y otra no, se devuelve la que la ha generado. Si en ninguna de las dos se ha logrado reconocer ningún carácter, se considera que la *CaptionBox* no es válida o no encierra texto y se ignora.

En caso de haber un resultado válido, éste se almacena en los descriptores del vídeo que servirán para generar toda la salida del *IVOnLA* y se continúa con la siguiente *CaptionBox*. Cuando se acaban las *CaptionBoxes* de un fotograma, se espera hasta que se localice otro fotograma con texto y se reinicia el proceso hasta que se termina de analizar todo el vídeo.

5 Pruebas y resultados

5.1 Base de datos de prueba

Para efectuar las pruebas que permiten evaluar los resultados del algoritmo se han seleccionado una batería de rótulos de texto superpuesto, obtenidos de distintos programas de cadenas tanto nacionales como internacionales, de programas de actualidad, noticiarios, programas de actualidad deportiva, etc. Muestran una diversidad de idiomas que comprenden castellano, inglés, francés y alemán, y contienen diversos tipos de fondo y contrastes, así como diferentes tipos de letras. Las fuentes de las que proceden son tanto de alta calidad (obtenidas de una señal de televisión SDTV) como de televisión retransmitida vía internet, que presentan una calidad muy baja. En total, el conjunto de datos consta de 92 imágenes de alta calidad, que contienen en total 369 palabras y 2119 caracteres y 39 de baja calidad, en las que se cuentan 136 palabras y 849 caracteres.

El algoritmo no requiere ningún tipo de entrenamiento ni es adaptativo en ningún sentido, por lo que las pruebas se han realizado directamente con toda la base de datos.

5.2 Realización de las pruebas

En sistemas de este tipo, las pruebas que se realizan tienen fundamentalmente que ver con la cantidad de caracteres y palabras correctamente segmentados y reconocidos. En los sistemas completos, para que una prueba se considere exitosa, las etapas de localización, segmentación y reconocimiento tienen que haber obtenido resultados satisfactorios. A continuación se describen los indicadores de calidad más comúnmente utilizados en la literatura, y posteriormente se detallan los resultados obtenidos por este algoritmo sobre la base de datos descrita en el apartado 5.1.

5.2.1 Indicadores de calidad

Las pruebas efectuadas incluyen el recuento del número de caracteres totales impresos en los rótulos, el número de caracteres correctamente reconocidos de entre éstos, y el mismo procedimiento con las palabras. Estos criterios coinciden con los típicamente aportados por otros investigadores del mismo campo y muestran las tasas de reconocimiento de palabras y caracteres, es decir, el porcentaje de palabras y caracteres que el algoritmo es, en media, capaz de segmentar y reconocer

correctamente. Si una palabra tiene un carácter mal reconocido, se asume que la palabra entera ha sido erróneamente reconocida.

CRR: Tasa de reconocimiento de carácter (*Character Recognition Rate*) es el porcentaje de caracteres reconocidos, dado por:

$$CRR = \frac{Número\ de\ caracteres\ reconocidos}{Número\ \ total\ de\ caracteres}$$

WRR: Tasa de reconocimiento de palabra (*Word Recognition Rate*) es, análogamente, el porcentaje de palabras reconocidas correctamente, calculado como:

$$WRR = \frac{Número\ de\ palabras\ reconocidas}{Número\ total\ de\ palabras}$$

Debido a la disparidad producida por la influencia de la calidad de las dos clases de imágenes utilizadas, los resultados se han separado para cada una de estas clases de imágenes.

5.2.2 Resultados

A continuación se muestran los resultados cuantitativos obtenidos al ejecutar el algoritmo sobre los dos conjuntos de prueba contemplados (alta y baja calidad), mediante los indicadores de calidad especificados en el apartado anterior.

Prueba 1: Batería de *captions* obtenida directamente de la señal de televisión (alta calidad):

$$CRR = \frac{2030}{2119} = 0.957$$

$$\text{WRR} = \frac{339}{369} = 0.919$$

Prueba 2: Batería de *captions* obtenida de televisión retransmitida vía internet (baja calidad):

$$CRR = \frac{544}{849} = 0.640$$

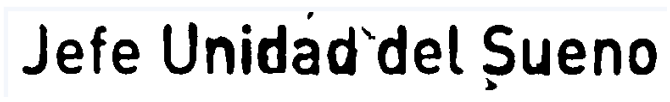
$$\text{WRR} = \frac{77}{136} = 0.566$$

A continuación se muestran los resultados cualitativos obtenidos sobre unas captions de ejemplo con diferentes tipos de fondos complejos y contrastes. En ellas se puede apreciar el aislamiento del texto y eliminación del fondo.


Tomás Rodríguez Bolaños


Tomás Rodríguez Bolaños


Jefe Unidad del Sueño

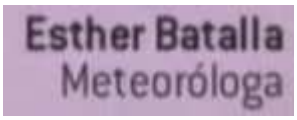

Jefe Unidad del Sueno


bbcworldnews.com/contactus


bbcworldnews.com/contactus


NATIONAL HOCKEY LEAGUE


NATIONAL HOCKEY LEAGUE


Esther Batalla
Meteoróloga


Esther Batalla
Meteoróloga


Road to the White House


Road to the White House

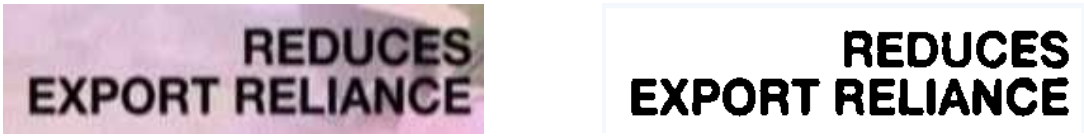


Figura 5.2.1 Ejemplos de resultados sobre rótulos de prueba.

5.2.3 Consideraciones sobre eficiencia

Se ha medido el tiempo que tarda el algoritmo en segmentar un carácter, y su comparación con el tiempo de reconocimiento empleado por el OCR. Las pruebas se han realizado sobre el siguiente equipo:

- Procesador: Intel® Core™ 2 duo T7500 2x2.20 GHz.
- Memoria RAM: 2.00 GB
- Tipo de sistema: 32 bits
- SO: Windows Vista™ Home Premium
- Entorno: Microsoft Visual C++ 2005 V 8.0

El tiempo medio que se emplea en la segmentación de un carácter, sin tener en cuenta el retardo introducido por el proceso de reconocimiento, es de 0.0073 segundos, o lo que es lo mismo, el sistema puede extraer aproximadamente 137.14 caracteres por segundo.

El tiempo invertido en segmentación va desde decenas de milisegundos, hasta pocas centenas. En cambio, el OCR requiere desde varias décimas de segundo hasta varios segundos para reconocer. El ratio entre ambos tiempos, calculado en varios rótulos de prueba es 19.8. Es decir, el tiempo invertido en el reconocimiento es casi veinte veces superior al que se tarda en segmentar.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este proyecto se ha desarrollado un algoritmo que permite la extracción de caracteres incrustados en rótulos en imágenes y vídeos MPEG, con un núcleo basado en la aplicación de operaciones morfológicas. Los resultados obtenidos avalan la utilización de estas técnicas, a las que hasta la fecha se les ha prestado escasa atención.

El mejor o peor funcionamiento para un rótulo particular dependerá de sus características propias, como la calidad o el contraste suficiente entre todos los caracteres y el fondo, que son las que facilitan la correcta segmentación y reconocimiento. También se debe tener en cuenta la uniformidad del texto dentro del mismo rótulo, y la uniformidad en el contraste (esto es, que o bien todo el texto sea claro o bien oscuro).

En cuanto al reconocimiento, probablemente sea posible aumentar la tasa de acierto sustituyendo el *OCR* de código libre del que nos hemos servido en este proyecto por algún otro software comercial, debido a las mejores prestaciones que estos programas suelen ofrecer con respecto a los gratuitos. No obstante, no se han realizado pruebas con ningún otro software que permitan precisar el impacto que podría tener el cambio de programa de reconocimiento.

Por otra parte, como se observó en el capítulo de resultados, la velocidad de ejecución del proceso se ve ampliamente limitada por el tiempo de reconocimiento en contraposición con el tiempo de segmentación, y por tanto las aplicaciones en tiempo real dependen casi exclusivamente de que se logre optimizar ese tiempo.

Los resultados de este trabajo se han enviado en forma de contribución científica a la *International Conference on Image Processing*, ICIP'2009. En el Anexo A se reproduce una copia íntegra de dicha contribución. Los resultados del proceso de evaluación de la citada conferencia se conocerán en junio de 2009.

6.2 Trabajo futuro

La eficacia de los métodos de extracción de texto viene en gran medida determinada por la calidad de los medios que se procesan. En relación a esto, podemos discernir dos vertientes que ejercen gran influencia sobre lo que debe ser este campo en el futuro. Por un lado, los sistemas de codificación son cada vez mejores y permiten

transmitir contenidos con menos bits, a veces a costa de la calidad de las imágenes. Si la calidad no llega a un límite aceptable, las mejoras deben ser dirigidas al campo del reconocimiento. Hasta ahora, los programas de reconocimiento, OCRs, tenían la función de traducir el texto de imágenes a códigos, pero para avanzar en este ámbito, debemos dotarlos de funciones avanzadas de comprensión lingüística e interpretación, de forma que sean capaces de "inferir" el mensaje de un texto defectuoso o incompleto. Por otro lado, los medios de transmisión y la capacidad de procesamiento de los ordenadores son cada vez mejores, lo cual permite transmitir medios de alta calidad. En este caso, los avances se pueden introducir en los algoritmos de segmentación, lo que mejorará los resultados sin ir en detrimento de una eficiencia aceptable. El compromiso entre estas dos vertientes supone el siguiente paso a dar en este campo.

Referencias

[1] Gllavata, J.; Ewerth, R.; Freisleben "A text detection, localization and segmentation system for OCR in images", IEEE Sixth International Symposium on B.Multimedia Software Engineering, 2004. Proceedings. Volume , Issue , 13-15 Dec. 2004 Page(s): 310 - 317

[2] Qixiang Ye; Wen Gao; Qingmig Huang "*Automatic text segmentation from complex background*" Image Processing, 2004. ICIP apos; 04. 2004 International Conference on Volume 5, Issue , 24-27 Oct. 2004 Page(s): 2905 - 2908 Vol. 5

[3] Rainer Lienhart, Axel Wernicke "*Localizing and Segmenting Text in Images and Videos*" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 2, NO. 4, APRIL 2002

[4] Jean-Marc Odobez and Datong Chen "*Robust video text segmentation and recognition with multiple hypotheses*" International Conference on Image Processing. 2002. Proceedings. 2002 On page(s): II-433- Ii-436 vol.2. Switzerland.

[5] Yaowen Zhan, Weiqiang Wang, Wen Gao. "*A Robust Split-and-Merge Text Segmentation Approach for Images*" 18th International Conference on Pattern Recognition, 2006. ICPR 2006. Volume: 2, On page(s): 1002-1005. Hong Kong.

[6] Datong CHEN. "*Text detection and recognition in images and video sequences*" Ph.D. Thesis . École Polytechnique Fédérale de Lausanne. infoscience@epfl.ch

[7] Teo Boon Chen; Ghosh, D. Ranganath, S. "*Video-text extraction and recognition*" TENCON 2004. 2004 IEEE Region 10 Conference Volume A, Issue , 21-24 Nov. 2004 Page(s): 319 - 322 Vol. 1

[8] Malobabić, Jovanka and O'Connor, Noel E. and Murphy, Noel and Marlow, Seán (2004) "*Automatic detection and extraction of artificial text in video".* In: WIAMIS 2004 - 5th International Workshop on Image Analysis for Multimedia Interactive Services, 21-23 April 2004, Lisboa, Portugal.

[9] D. Karatzas and A. Antonacopoulos "*Two Approaches for Text Segmentation in Web Images*" PRImA Group, Department of Computer Science, University of Liverpool, Peach Street, Liverpool L69 7ZF, United Kingdom http://www.csc.liv.ac.uk/~prima

[10] C. Garcia and X. Apostolidis. "*Text detection and segmentation in complex color images*" IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 Volume: 6, On page(s): 2326-2329 vol.4. 2000. Istanbul, Turkey.

[11] Nobuyuki Otsu, "*A threshold selection method from gray-level histogram*", IEEE Transactions on System Man Cybernetics, Vol. SMC-9, No. 1, 1979, pp. 62-66.

[12] D2.8: MESH Content Analysis Modules, consolidated and integrated versions". Technical report, The MESH project.2009

Glosario

GMM	Gaussian Mixture Model
JPEG	Joint Photographic Experts Group
OCR	Optical Character Recognition
CRR	Character Recognition Rate
WRR	Word Recognition Rate
DCT	Discrete Cosine Transform
SDTV	Standard Definition Television

Anexos

Anexo A

Artículo de título "*Morphological approach to caption segmentation and recognitino in video frames"* que ha sido presentado en forma de contribución científica a la International Conferenece on Image Analysis (ICIP 2009), y cuya aceptación será conocida en junio del 2009.

MORPHOLOGICAL APPROACH TO CAPTION SEGMENTATION AND RECOGNITION IN VIDEO FRAMES*

Juan Iglesias, Jesús Bescós

Video Processing & Understanding Lab, Universidad Autónoma de Madrid (Spain)
e-mail: Juan.Iglesias@estudiante.uam.es, J.Bescos@uam.es

ABSTRACT

This paper presents an especially efficient technique to extract binary text masks from typical caption regions present in TV video signals, both in high quality SDTV and in low quality TV over IP. Combined with an open source OCR it achieves character and word recognition rates that outperform those reported to date. The strengths of the technique are its novel simplicity and robustness: it is based on a morphological top-hat operation, which just requires a previous estimation of the characters' stroke width. Later resizing and connected components post-processing to consider typical character restrictions almost follow state-of-the-art approaches.

Index Terms— video analysis, caption extraction, caption recognition, text recognition, OCR.

1. INTRODUCTION

Text localization and segmentation in images and videos with complex background is one of the key features to be considered for information selection in the massive video databases available. Therefore, lots of research has been devoted to develop successful methods.

The most common ways to face this problem include image splitting into two or more new images or layers, each covering some gray levels; pixel clustering, considering characteristics such as color, luminance, location, etc., to model the text pixels; and region growing schemes that discard areas until the text areas as isolated. Connected component analysis is a common post processing approach to remove errors according to a-priori known text restrictions. Whatever the method, multiple hypotheses or binary masks can be generated by varying some of its parameters, so that different outputs are available for ultimate decision. Finally, an OCR results or other simpler criteria is applied to select the best output produced.

The work in [1] estimates the background and text color by histogram subtraction and then uses unsupervised k-means over a feature vector considering RGB values and the standard deviation of wavelet coefficients in the three high frequency sub-bands. This is motivated by the fact that text characters usually share texture and also show high contrast edges. In [2], they use both the Euclidean distance and a cosine-based similarity to cluster pixels into text or non-text. Then a Log-Gabor filter is used in order to decide which of the clusters created better fits with the text layer. Pixel clustering methods are computationally intensive, and also have the problem of deciding how many clusters should a certain image be split into.

In [3] the authors report a multi-layer approach which clusters pixels into different layers and then forwards each to an OCR, afterwards deciding which layer best fits with the text. These kind of approaches are computationally expensive, since not all images can be divided into the same number of classes, which leads to the need of creating a large number of possibilities to find the right one. They also require several calls to an OCR, which is the most time-consuming module in segmentation and recognition systems. In [4] they train a Gaussian Mixture Model with pixels from the interior of some characters in the image to create an accurate model of the text color and then cluster pixels. They begin with one model and then iteratively increase this number by testing the number of recognized characters, which results in some sort of multi-layer approach. Then, spatial connectivity and connected component analysis is performed in order to enhance the result before finally forwarding a binary image to the OCR.

The work in [5] describes the use of a *SeedFill* algorithm to remove areas of the same color of the outer regions of the bounding box, trying to isolate the inner characters. This works poorly with modern text insertion techniques that perform anti-aliasing in character's edges.

None of the referenced approaches meet the requirements of real-time operation. We here present an unsupervised caption recognition method focused on very simple and efficient techniques, which reduces the post-processing burden typically required to make the caption-mask suitable for an OCR. Only horizontal text is considered, since it covers the majority of the text captions on videos and images. The presented algorithm has been integrated within a real-time caption extraction method [6], which extracts frame regions containing captions in MPEG video.

The next section shows an overview of the approach and then deepens into each stage. Comparative results are then presented in section 3 and section 4 concludes the paper.

* Work supported by the Spanish Government (TEC2007-65400 - SemanticVideo), the European Commission (IST-FP6-027685 - Mesh) and the Comunidad de Madrid (S-0505/TIC-0223 - ProMultiDis-CM).

2. ALGORITHM DESCRIPTION

Fig. 1 shows a diagram of the algorithm flow. After converting the input image (in fact an image region defined in a video frame via a bounding box) to a gray-level image, the algorithm starts estimating the width of the character stroke by using projection profile information. This estimation controls the dimension of the structuring element used to perform a morphological top-hat The resulting image is then thresholded to obtain a binary image. A simple connected component analysis is then performed to remove image regions that are not likely to be characters due to their size or shape. Finally, a binary image is forwarded to an OCR module.

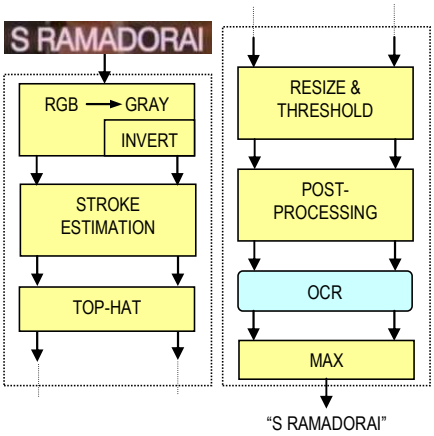


Figure 1*: Algorithm overview.*

The core of the algorithm performs a top-hat which performs the desired operation if text areas represent maxima, but if they represent minima. As no robust and simple way has been found to a-priori decide on text being maxima or minima, the algorithm is performed twice for each caption image: first to the original image and, in parallel, to the inverted one. One of the OCR results will be clearly superior to the other.

Color information has been discarded, as it slowed down all the operations while almost not providing great extra information compared to the grayscale image, particularly when focusing on morphological operations instead of on classification approaches.

2.1. Character stroke estimation

As aforementioned, the essence of this method resides in the use of the morphological top-hat operation for text segmentation. Taking advantage of the fact that all the characters in a caption usually share a single font type and therefore a specific width of the character's strokes, they can be segmented by morphological means. So, the first step to be done is the estimation of the character's stroke.

It is crucial for the accuracy of this stage to exactly locate a text line inside the bounding box. In order to achieve this, the original image is filtered with a Canny edge detector so that all the character's edges and other edges are marked with a single pixel contour. Then, the horizontal projection profile is calculated (see Fig. 2). The profile is scanned until two consecutive row-values are located over the mean; starting from these two rows, the top and bottom limits of the text line are identified as the first connected previous and following row, respectively, whose values fall below half the mean.

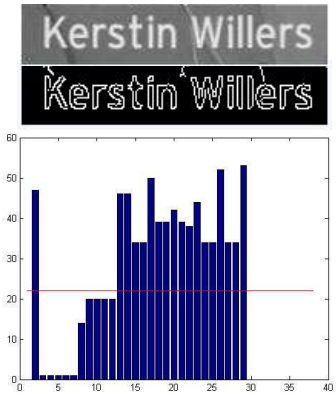


Figure 2*: Original caption image (top), image after Canny edge detector (middle) and horizontal projection profile (bottom), with the mean value remarked..*

Once the text-line pixel rows are identified, a vertical projection profile is calculated over the part of the original grayscale image limited by them. The profile shows peaks whose width can statistically characterize the character's stroke (see Fig. 3). However, not all the characters are suitable for this operation, and so the method's success depends on the existence of enough peaks belonging to appropriate characters such as 'I', 'j', 'l', 'd', 'f', etc. Other characters such as 'w' or 'x' do not present a projection profile from which the width of its stroke can be precisely obtained. To obtain a robust estimation, the mean width was estimated discarding values two standard deviations away from the initial mean.

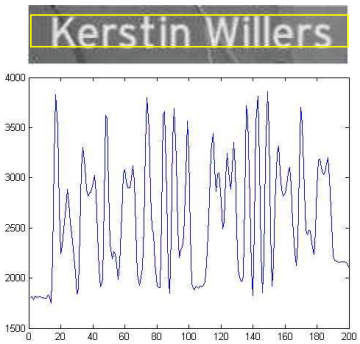


Figure 3*: Vertical projection profile of a text line.*

2.2. Text mask extraction

The background variability in image-inserted captions causes most traditional text segmentation approaches (e.g, gradient evaluation, region growing) to miss parts of the characters that look similar to parts of the background, unless complex post-processing or decision schemes are implemented. This results in significant degradation of an OCR performance. The morphological top-hat operation is a simple and efficient way to confront this situation.

A crucial aspect in morphological operators is the selection of the structuring element, both its shape and its size. A diamond-shaped morphological structuring element has proven to be the best containing the character's parts, mainly due to the stroke of the most commonly used fonts. However, almost any other shape such as a circle or a square works just fine. With regard to the size, the structuring element should be slightly larger than the estimated stroke width, so that *the hat covers* the maximums represented by the characters' stroke. Our tests over a broad set of captions suggest that the element should be at least 5 or 6 pixels larger than the estimated width, so as to avoid it fitting into wider character areas.

Although caption images are conceptually bimodal (the text and the rest) their histogram is usually far from being bimodal. The top-hat operation produces an image histogram much more bimodal that the original one, hence enhancing the performance of a thresholding technique (see Fig. 4). This allows the algorithm to use the simple Otsu's method for this task.

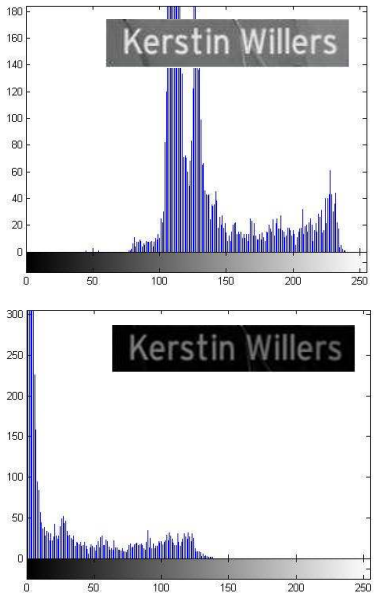


Figure 1*: Histogram of the original image (top) and of the top-hat image (bottom).*

Common OCRs are designed to work with images scanned with a resolution of about 300 dpi, far from typical caption images resolution. Hence, increasing the image resolution before forwarding it to the OCR significantly improves the recognition ratio. In this line, prior to binarization, resolution is proportionally increased, via cubic interpolation, to a height of 100 pixels per text line. It has been tested that, for the used OCR, higher resolution, while implying higher computational cost, does not result in a better recognition ratio.

2.3. Text mask post-processing

The processing stages described up to now produce a text mask which may eventually include non character areas. However, as characters should fulfill certain shape restrictions, which can be normalized for a 100 pixels text line, a simple connected components techniques can help discarding such areas. In this line, we have compiled and slightly modified heuristics reported by some authors [3][7], so that a connected component is removed if:

- It is bigger than 30 pixels, which is the minimum size of a dot in an 'i' or a 'j'.
- Its width/height ratio is between 2.5 and 0.1, which is set according to the characters "i" and "W" in different fonts.
- Its width is less than 2.1 the height of the whole text region, which is also set according to the character "W" in different fonts.
- Its compactness is lower than 0.2. This means that at least 20% of the pixels of a character's bounding box, must belong to the character.

Finally, the resulting text mask is inverted prior to feed the OCR. Fig. 5 shows some results of the overall text extraction stage.

Figure 2*: Caption images and the resulting text mask*

3. EXPERIMENTS AND RESULTS

The presentation of reliable comparative results is not an easy issue, due to the lack of a public caption database and to the diversity of existing caption types. Moreover, reported works provide different kind of quality indicators to illustrate their results.

The Character Recognition Rate (CRR) and the Word Recognition Rate (WRR) are two of the most commonly used indicators. They respectively measure the rate between the number of correctly recognized characters or words and the total number of them. One word is considered correctly recognized if all its characters have been correctly recognized. The tricky question is the OCR: while being an external part of the referenced techniques, clearly determines these quality indicators. Its influence in the final results is hard to evaluate.

The presented approach has been tested on a database of 92 high-quality (HQ) captions obtained directly from digital TV video (including 369 words and 2119 characters), and 39 low-quality (LQ) captions coming from screenshots of internet-broadcasted TV (including 136 words and 849 characters). The selected OCR is Tesseract V.2.03, an open source tool available in http://code.google.com/p/tesseract-ocr/.

Despite the aforementioned difficulties, in order to provide some kind of comparative objective measure, Table 1 reproduces some representative reported results. In [1], they refer to a test set of 38 images obtained from commercial TV broadcasts, web images, MPEG-7 videos and movies with 325 words and 2156 characters. Results in [8] just include the CRR obtained for a data set consisting of captions obtained from domestic and broadcasted video sequences, and images from the WWW. Finally [5] reports results over a set of 100 images coming from web pages and 300 video frames, embedding 4861 Chinese words.

	WRR	CRR
Our approach (HQ)	91.9%	95.7%
Our approach (LQ)	56.6%	64%
Algorithm in [1]	55.83 %	77.71 %
Algorithm in [5]	Not provided	69.9%
Algorithm in [4]	85.5% (Chinese)	Not provided

Table 1*: Algorithm results over the test set and results reported by other studies.*

Regarding efficiency, the mean time spent in the text mask generation process (not considering the delay due to recognition tasks) is 0.0073 seconds or, in other words, the algorithm is able to extract 137.14 characters per second.

4. CONCLUSIONS

This paper has presented a method to automatically extract text-masks from horizontal captions embedded in complex backgrounds, by means of morphological operations. These have proved to provide a simple and robust solution, compared to most reported work based on gradients evaluation, pixel clustering, etc.

In combination with an open source OCR, the presented technique achieves results significantly superior to those reported by other algorithms with the same target, always keeping in mind the objections to compare results using different OCR modules. In addition to this, its real-time working capabilities are noteworthy, which is a critical issue when fast information indexing over massive video databases is intended.

Performed tests have also proved that morphological processing is an approach suitable for a vast majority of caption types currently appearing in digital media.

Further research should consider merging the segmentation and recognition stages, so that common character shapes and characteristics are available for the segmentation to be more precise. Also the use of OCRs with semantic and linguistic features that allow them to complete words from some of their characters will significantly improve overall recognition performance.

REFERENCES

[1] J. Gllavata, R. Ewerth, B. Freisleben, “A text detection, localization and segmentation system for OCR in images”, Proc. IEEE Intl. Symposium on Multimedia Software Engineering, 2004.

[2] C. Mancas-Thillou, B. Gosselin, “Spatial and Color Spaces Combination for Natural Scene Text Extraction”, Proc. IEEE Intl. Conference on Image Processing, 2006.

[3] D. Chen, J-M. Odobez, H. Bourlard, “Text detection and recognition in images and video frames”. Pattern recognition, vol. 37, nº 3, March 2004.

[4] Q. Ye., W. Gao, Q. Huang, “Automatic text segmentation from complex background”, Proc. IEE Intl. Conference on Image Processing, 2004.

[5] R. Lienhart, A. Wernicke “Localizing and Segmenting Text in Images and Videos”, IEE Transactions on Circuits and Systems for Video Technology, vol 2, nº 4, April 2002.

[6] D. Márquez, J. Bescós, “A Model-based Iterative Method for Caption extraction in Compressed MPEG Video”, Proc. Intl. Conference on Semantic and Digital Media Technologies (SAMT’07), Lecture Notes in Computer Science, vol. 4816, 2007.

[7] D. Karatzas, “Text Segmentation in Web Images Using Color Perception and Topological Features” PhD disertation, University of Liverpool (UK), 2003.

Anexo B

Bases de datos completas sobre las que se ha testado el algoritmo y se han obtenido los resultados.

Base de datos de imágenes obtenidas directamente de una señal de televisión (SDTV).

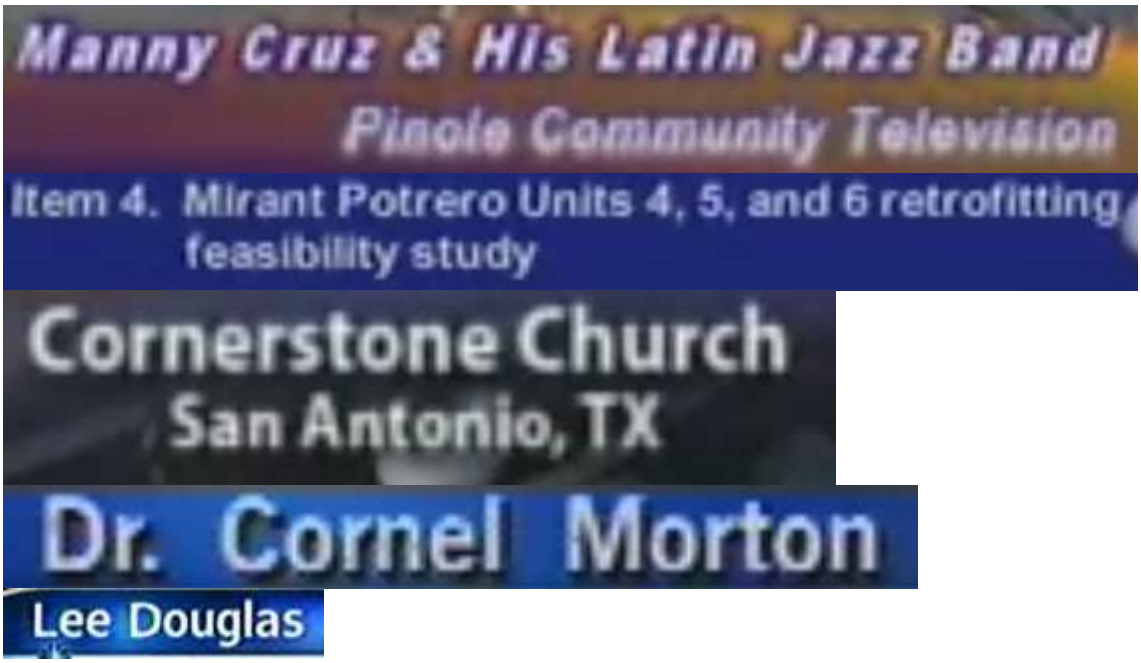

AGREDIDA POR OTRA MENOR
Palacio de la Moncloa (Madrid)
euronews.net
markets
QUENTIN SOMMERVILLE
Beijing drivers to leave car at home once a week
Road to the White House
LEADING OUR TROOPS
REDUCES
EXPORT RELIANCE
ENGLAND REFUSE MADRID FRIENDLY
DETENIDO UN PRESUNTO
VIOLADOR EN SERIE
VENEZUELA
Imágenes captadas por la cámara
Experta en tribunales agencia Fax Press
LA SEDA BARCELONA
ECONOMIA
CONTINUO
diferencia
Tomás Rodríguez Bolaños
Jefe Unidad del Sueño
bbcworldnews.com/contactus
WARSAW
STOCK
EXCHANGE
IN 1999,HAIDER'S FREEDOM PARTY'S
COLOMBEY-LES-DEUX-EGLISES
SARKOZY AND MERKEL SHOW UNITED FRONT


ZIMBABWE'S MUGABE APPOINTS KEY MINISTERS
U.S. REMOVES N.KOREA FROM TERRORISM BLACKLIST
THE SHOCK OF THE LIGHTNING
Dig out your soul
Postcards From Paradise
Sardinia
FINANCIAL CRISIS
No, and if you read the report you'll see that
FINANCE MINISTERS, INCLUDING
Raymond DOMENECH
EBAY, JOHNSON&JOHNSON, UNITED TECH
that I want to take home with me.
Das ist schwer zu toppen.
DAVID CARRO
S RAMADORAI
www.foxnews.com
POLICY CRITICISM OF OBAMA
EVERY THURSDAY
Washington 10.10 am
MUSIC AT THE ABBEY
NATIONAL HOCKEY LEAGUE
JAPAN SAYS LIST RECALL
LUÍS MANTIÑÁ LÓPEZ
Aragoneses
TAMBIÉN DISPONIBLE EN TU CENTRO
CARREFOUR MÁS CERCANO
TV5MONDE DANS UN INSTANT
Whatever your moment, Allianz
gives you the confidence you need.

Base de datos de imágenes de baja calidad obtenidas de señales de televisión retransmitida por internet.


N.Y. Mets
Adhar Hakim
Reporter SCTV di depan Hotel Marriot
Steuerschlupflöcher für Unternehmen schließen
Frankfurt
Maria da Conceição
73 anos
Ana Blanco
José María Aznar
PRESIDENTE DEL GOBIERNO
Actríz Invitada
Miriam Díaz Aroca
April Madison
OKs Teachers Packing Heat
BUILDING FIRE
South L.A.
TUES 10AM
EARLIER
SHERRY BEBITCH JEFFE
POLITICAL GURU
Glenn Gordon Caron
JUEVES 28 de AGOSTO
18:00 horas (DIRECTO)
COMISIÓN DE ECONOMÍA Y HACIENDA
IBEX 35 9%
Los instantes posteriores


Lois Wolk
Cmte. Chair
California State Assemblymember
District 8, Davis -D
BREAKING+NEWS
CALABASAS FIRE
BRUSH FIRE; LAS VIRGINES & LOST HILLS;
LIGHT GRASS; NO WIND
Republican National Committee Video
REPUBLICAN PARTY PLATFORM
Republican National Committee
Jerry Springer
Jerry Springer
HOST, "AMERICA'S GOT TALENT"
Trish Kelley
Mayor
Rick Phinney


Manny Cruz & His Latin Jazz Band
Pinole Community Television
Item 4. Mirant Potrero Units 4, 5, and 6 retrofitting feasibility study
Cornerstone Church
San Antonio, TX
Dr. Cornel Morton
Lee Douglas

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido).....2.000 €
- Material de oficina.....150 €
- Total de ejecución material .....2.150 €

2) Gastos generales

- 16 % sobre Ejecución Material .....344 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material .....129 €

4) Honorarios Proyecto

- 960 horas a 15 € / hora .....14400 €

5) Material fungible

- Gastos de impresión .....200 €
- Encuadernación .....60 €

6) Subtotal del presupuesto

- Subtotal Presupuesto .....16810 €

7) I.V.A. aplicable

- 16% Subtotal Presupuesto .....2689.6 €

8) Total presupuesto

- Total Presupuesto.....19499.6 €

Madrid, Marzo de 2009

El Ingeniero Jefe de Proyecto

Fdo.: Juan Iglesias Cavada
Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de extracción y reconocimiento de rótulos en vídeos MPEG. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego. Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.
4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el
contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda

exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.