



Escuela  
Politécnica  
Superior

# DETECCIÓN E IDENTIFICACIÓN VISUAL DE CARACTERES EN PRODUCTOS INDUSTRIALES



Máster Universitario en  
Automática y Robótica

## Trabajo Fin de Máster

Autor: Juan José Ortuño López

Tutor: Fernando Torres Medina

Julio 2016



Universitat d'Alacant  
Universidad de Alicante



## Indice de Contenidos

I. Justificación y Objetivos .....	5
II. Agradecimientos .....	5
III. Dedicatoria .....	5
IV. Citas.....	6
V. Índice de Ilustraciones .....	6
1. Introducción .....	7
2. Marco Teórico .....	7
2.1. Detección de caracteres.....	7
2.1.1. La imagen digital .....	7
2.1.2. Conversión a escala de grises.....	9
2.1.3. Preprocesamiento de la imagen .....	10
2.1.3.1. Filtro de media .....	10
2.1.3.2. Filtro de mediana .....	11
2.1.4. Histograma de una imagen digital .....	12
2.1.5. Binarización de una imagen en escala de grises .....	14
2.1.6. Segmentación.....	14
2.1.6.1. Segmentación basada en píxeles .....	14
2.1.7. Métodos de binarización.....	15
2.1.7.1. Binarización mediante el método de Otsu.....	15
2.1.7.2. Binarización mediante la media.....	16
2.1.8. Filtro de tamaño.....	16
2.1.9. Etiquetado y Cuenta de objetos.....	17
2.2. Identificación y reconocimiento de caracteres .....	18
2.2.1. Descripción y extracción de características .....	18
2.2.1.1. Introducción .....	18
2.2.1.2. Extracción de características .....	19
2.2.1.3. Descriptores .....	20
2.2.1.3.1. Descriptores de forma simple .....	20
2.2.1.3.2. Envoltentes .....	21
2.2.1.3.3. Distancias de signature o curvas .....	21
2.2.1.3.4. Momentos Generales.....	21
2.2.1.3.5. Momentos Invariantes .....	22

2.2.1.3.5.1. Momentos invariantes a traslaciones .....	22
2.2.1.3.5.2. Momentos Invariantes a Homotecias .....	23
2.2.1.3.5.3. Momentos invariantes a Traslaciones, Rotaciones y Homotecias .....	24
2.2.2. Reconocimiento de caracteres.....	25
2.2.2.1. K-NN .....	25
2.2.2.2. Árboles de decisión .....	26
2.2.2.3. Redes neuronales .....	26
3. Objetivos.....	27
4. Metodología .....	27
5. Descripción de la solución .....	27
5.1. Definición del algoritmo propio de detección.....	27
5.1.1. Descripción del algoritmo.....	27
5.1.2. Pruebas realizadas.....	30
5.1.2.1. Pruebas con el método de Otsu .....	30
5.1.2.2. Pruebas realizadas con el algoritmo desarrollado .....	32
5.2. Definición del algoritmo propio de reconocimiento .....	44
5.2.1. Descripción del algoritmo.....	44
5.2.2. Pruebas realizadas.....	47
6. Conclusiones.....	53
7. Bibliografía y Referencias .....	54
8. Anexos .....	55

## **I. Justificación y Objetivos**

El reconocimiento óptico de caracteres (OCR, Optical Character Recognition) es esencial en el tratamiento de imágenes para la identificación de secuencias de caracteres con un significado concreto.

Desde la aparición de los algoritmos de OCR, han sido muchos los servicios que han introducido estos procesos para aumentar su rendimiento y otros que se basan completamente en estas tecnologías. Por ejemplo, el reconocimiento de texto en imágenes o en productos industriales, reconocimiento de texto manuscrito, reconocimiento de matrículas, reconocimiento de datos estructurados, son sólo algunos ejemplos dentro de otras muchas aplicaciones.

En la mayoría de ocasiones, los caracteres que se han de reconocer aparecen con colores o tonos que los hacen resaltar respecto del entorno que los acompaña, con lo que es tarea sencilla separarlos del resto de la imagen. Pero a veces no existe mucho contraste entre los caracteres que se han de identificar y el entorno o fondo sobre el que se presentan, por lo que a la tarea de reconocimiento se añade una tarea previa de detección de los caracteres que no se distinguen con claridad de su entorno.

El objetivo del presente trabajo consiste en el desarrollo de algoritmos en MATLAB para el reconocimiento automático de secuencias de caracteres a partir de imágenes digitales de productos industriales. Se trata de capturar una imagen de un producto industrial e identificar los caracteres de interés en la misma, que identifican determinadas características del producto. En las imágenes correspondientes a determinados productos, los caracteres podrán aparecer poco contrastados, lo que conlleva la dificultad añadida de su detección.

## **II. Agradecimientos**

Quiero expresar mi agradecimiento a mi tutor Dr. Fernando Medina Torres, por su apoyo y orientación durante la realización del presente trabajo.

A todos los profesores del máster, que han sabido impartir sus enseñanzas y transmitirnos una parte de su conocimiento.

A mi familia, a la que he robado muchas horas para afrontar este reto.

## **III. Dedicatoria**

A mi esposa Cristina, y mis hijas Cristina y Celia.

## IV. Citas

Cualquier tecnología suficientemente avanzada es indistinguible de la magia. Arthur C. Clarke.

## V. Índice de Ilustraciones

Figura 1. Adquisición de una imagen digital.....	7
Figura 2. Imagen digital .....	8
Figura 3. Canales RGB de una imagen en color .....	9
Figura 4. Respuesta del ojo humano al espectro visible .....	10
Figura 5. Filtro de media.....	11
Figura 6. Filtro de mediana.....	12
Figura 7. Histograma de una imagen en escala de grises .....	13
Figura 8. Histogramas de los canales RGB de una imagen en color.....	13
Figura 9. Binarización de una imagen según distintos niveles de umbral.....	14
Figura 10. Eliminación de objetos de pequeño tamaño.....	17
Figura 11. Etiquetado y cuenta de objetos .....	18
Figura 12. Fraccionamiento del histograma en diferentes rangos de niveles de gris.....	29
Figura 13. Aplicación método de Otsu en imagen con dos niveles de gris .....	30
Figura 14. Aplicación método de Otsu con varios niveles de gris en caracteres .....	30
Figura 15. Aplicación método de Otsu con varios niveles de gris en caracteres y en entorno.....	31
Figura 16. Aplicación método de Otsu en imagen de producto industrial .....	31
Figura 17. Aplicación método de Otsu en imagen de producto industrial .....	31
Figura 18. Algoritmo propio aplicado a imagen de producto industrial .....	33
Figura 19. Algoritmo propio aplicado a imagen de producto industrial .....	34
Figura 20. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (1).....	35
Figura 21. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (2).....	36
Figura 22. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (3).....	37
Figura 23. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (1) .....	38
Figura 24. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (2) .....	39
Figura 25. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (3) .....	40
Figura 26. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (1).....	41
Figura 27. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (2).....	42
Figura 28. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (3).....	43
Figura 29. Plantillas de caracteres de muestra para generar matriz de entrenamiento .....	44
Figura 30. Matriz de entrenamiento .....	45
Figura 31. Matriz de resultados.....	46
Figura 32. Determinación de endpoints y branchpoints para la letra A .....	48
Figura 33. Determinación de endpoints y branchpoints para la letra C .....	49
Figura 34. Determinación de endpoints y branchpoints para la letra Q.....	50
Figura 35. Reconocimiento de caracteres con forma original .....	51
Figura 36. Reconocimiento de caracteres esqueletizados .....	52

## 1. Introducción

En el presente trabajo se va a llevar a cabo el desarrollo de algoritmos en MATLAB que permitan la detección y la identificación de caracteres impresos o grabados en productos industriales. Se podrá dar el caso de que dichos caracteres aparezcan poco contrastados respecto de su entorno.

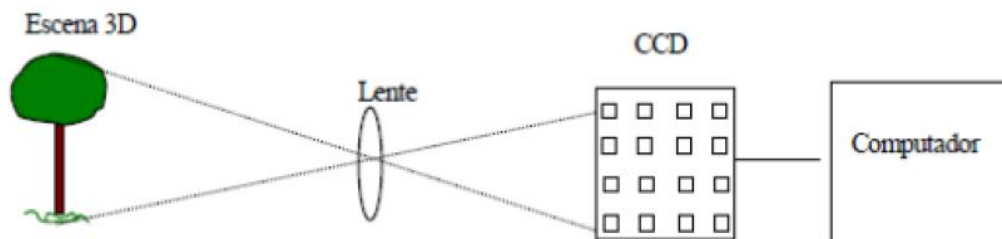
Consta de dos partes principales. Una de ellas es la correspondiente a la detección de los caracteres, incluso en el caso de existir poco contraste con su entorno. La otra parte es la de reconocimiento de los mismos y por tanto de su significado en el producto en el que aparecen.

## 2. Marco Teórico

### 2.1. Detección de caracteres

#### 2.1.1. La imagen digital

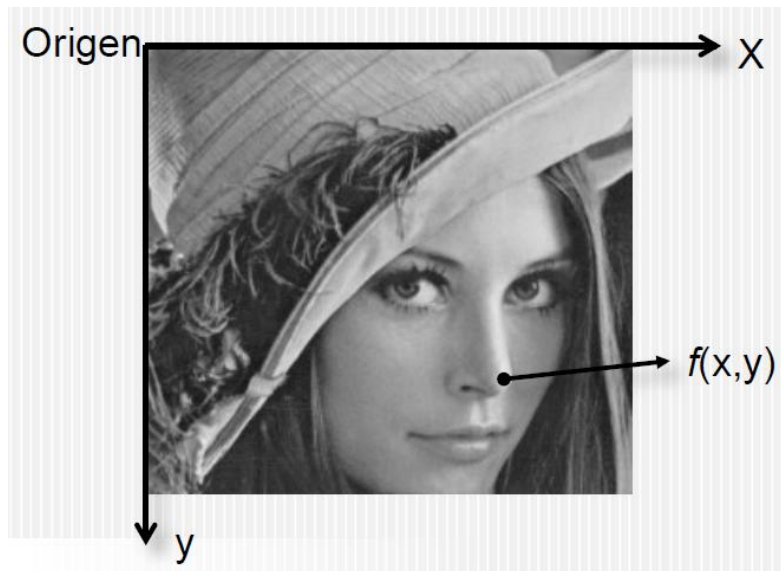
Las imágenes digitales son capturadas mediante los dispositivos apropiados para su almacenamiento en un ordenador. Estos dispositivos poseen una distribución en forma de matriz de elementos denominados píxeles. Cada elemento se activa con la incidencia de la radiación espectral, generando un valor de tensión eléctrica a la salida que se convierte a un valor numérico. Todos estos valores, organizados igualmente en forma de matriz se almacenan en el ordenador. Este almacenamiento se lleva a cabo a través de algún formato característico de imágenes (TIFF, BMP, JPEG).



**Figura 1. Adquisición de una imagen digital**

La imagen se representa desde el punto de vista de su tratamiento computacional como una matriz numérica de dimensión  $M \times N$ , es decir, con  $M$  filas y  $N$  columnas. El contenido de esa matriz se refiere a valores enteros situados en las localizaciones espaciales  $(x,y)$  o píxeles, dicho valor es el resultado de la cuantificación de intensidad o nivel de gris.

Si la imagen es en blanco y negro, se almacena un valor por cada píxel. Este valor es el nivel de intensidad o nivel de gris comentado anteriormente. Se suele utilizar un rango de valores para su representación, que generalmente es de 0 a  $2^n-1$ . Uno de los valores más utilizados es con  $n$  igual a 8; esto significa que el rango de valores para este caso varía entre 0 y 255. En este caso, el 0 representa el negro absoluto y el 255, el blanco absoluto.



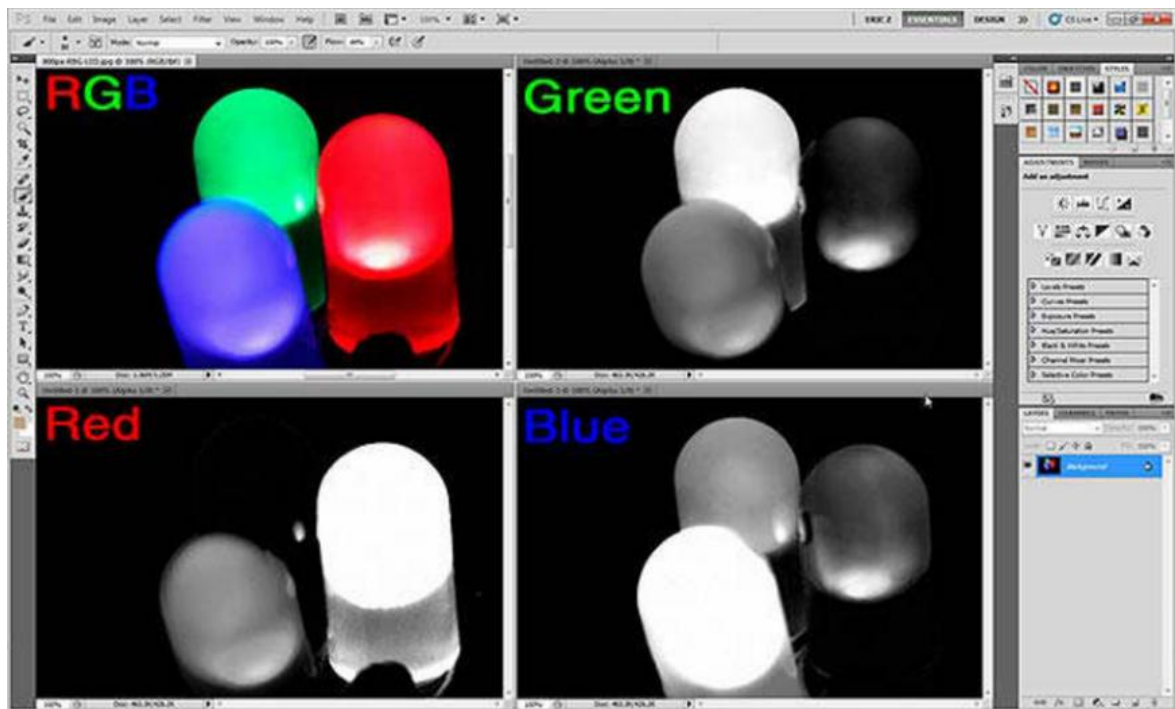
**Figura 2. Imagen digital**

Sobre cada matriz se establece un sistema de coordenadas con origen normalmente en la esquina superior izquierda y con los ejes  $x$  e  $y$  tales que su orientación es positiva hacia la derecha en el caso del eje  $x$  y hacia abajo en el caso del eje  $y$ .

Cuando la imagen es en color, para cada localización espacial existen tres valores de intensidad asociados, es decir, los elementos de la matriz vienen dados por tres valores que representan cada uno de los componentes básicos del color en cuestión. Estos componentes son el rojo (R), verde (G), y azul (B) y es lo que conocemos como código RGB. En este caso el conjunto de valores (0, 0, 0) representa al negro, mientras que los valores (255, 255, 255) es el blanco absoluto. La combinación de distintos valores representa otros colores. Debido a lo anterior, una imagen en color posee tres bandas espectrales: rojo, verde, azul; cada una de ellas viene representada por una matriz de números con valores en el rango 0 a 255 para imágenes de 8 bits. Una imagen de color vendría dada por las tres subimágenes de la Figura 3.

El píxel que se encuentra en la localización espacial  $(x,y)$  posee componentes (R, G, B). Los tres valores de cada píxel para las imágenes en color corresponden al modelo de color RGB.

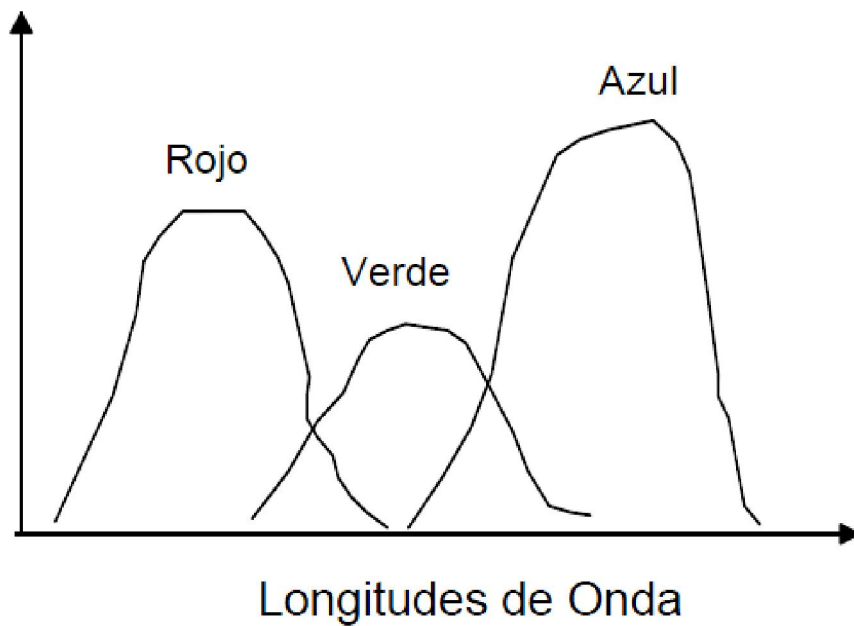




**Figura 3. Canales RGB de una imagen en color**

### **2.1.2. Conversión a escala de grises**

En esta parte se trata la conversión de una imagen en color a escala de grises, el equivalente a la luminancia de la imagen. Como sabemos el ojo percibe distintas intensidades de luz en función del color que se observe, esto es debido a la respuesta del ojo al espectro visible, la cual se puede observar en la Figura 4, por esa razón el cálculo de la escala de grises o luminancia de la imagen debe realizarse como una media ponderada de las distintas componentes de color de cada pixel.



**Figura 4. Respuesta del ojo humano al espectro visible**

La ecuación de la luminancia es la expresión matemática de ese fenómeno, y los factores de ponderación de cada componente de color nos indican la sensibilidad del ojo humano a las frecuencias del espectro cercanas al rojo, verde y azul.

$$Y = 0.3 \cdot R + 0.59 \cdot G + 0.11 \cdot B$$

Por tanto, para realizar esta conversión basta con aplicar la ecuación a cada pixel de la imagen de color, entonces resultará una nueva matriz de un byte por pixel que daría la información de luminancia.

### **2.1.3. Preprocesamiento de la imagen**

En la mayoría de ocasiones, antes de trabajar con una imagen digital, es conveniente mejorar la calidad de la misma, ya que se suelen generar imperfecciones en el proceso de captura de la misma.

Hay numerosas técnicas de filtrado con las que se consigue mejorar la calidad. Entre los más habituales se encuentran los siguientes.

#### **2.1.3.1. Filtro de media**

Para un filtro espacial de 3x3 (grado 3), la construcción más simple consiste en una máscara en la que todos los coeficientes sean iguales a 1/9. El valor de nivel de gris de cada pixel pasará a

ser la media de nueve valores (su nivel de gris y el de sus 8 vecinos). La Figura 5 muestra la máscara resultante.

$$g(x, y) = \frac{1}{P} \sum_{u=-r}^r \sum_{v=-r}^r f(x - u, y - v)$$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

**Figura 5. Filtro de media**

### **2.1.3.2. Filtro de mediana**

Una de las principales dificultades del filtro de media, es que afecta los bordes y otros detalles de realce. Cuando el objetivo es reducir el ruido, el empleo de los filtros de mediana representa una posibilidad alternativa. En este caso, el nivel de gris de cada pixel se reemplaza por la mediana de los niveles de gris en un entorno de este pixel, en lugar del promedio, como lo hace el filtro de media.

Este método es particularmente efectivo cuando el patrón de ruido consiste en componentes fuertes y de forma puntiaguda, y la característica que se desea preservar es la agudeza de los bordes.

La mediana  $m$  de un conjunto de valores es tal que la mitad de los valores del conjunto quedan por debajo de  $m$  y la otra mitad por encima. Con el fin de realizar el filtro de mediana, en el entorno de un pixel, primero se deben extraer los valores del pixel y de su entorno, determinar la mediana y asignar este valor al pixel. Por ejemplo, para un entorno de 3x3, se realizan los pasos que se pueden ver en la Figura 6.

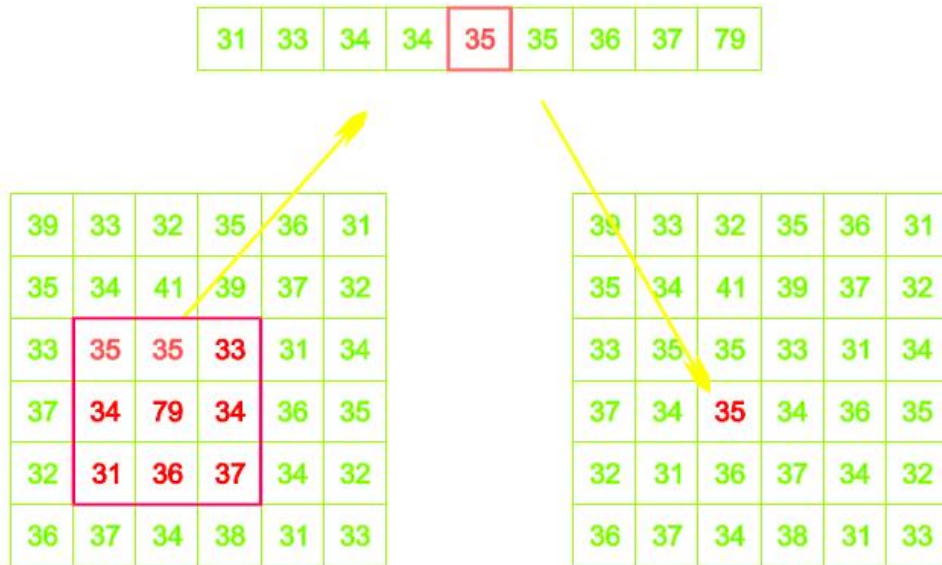


Figura 6. Filtro de mediana

### 2.1.4. Histograma de una imagen digital

El histograma de una imagen es una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad,  $g$ . La probabilidad de ocurrencia de un determinado nivel se define como:

$$P(g) = \frac{N(g)}{M}$$

Donde  $M$  es el número de píxeles en la imagen y  $N(g)$  es el número de píxeles en el nivel de intensidad  $g$ . Como con cualquier distribución de probabilidad todos los valores de  $P(g)$  son menores o iguales que 1 y la suma de todos los valores de  $P(g)$  es 1.

El histograma de una imagen y su representación constituye una herramienta visual de gran utilidad para el estudio de imágenes digitales, que puede proporcionar una idea muy aproximada de la distribución de niveles de gris.

Las intensidades o niveles de gris están representadas a lo largo del eje  $x$  y el número de ocurrencias para cada intensidad se representa en el eje  $y$ .

En la Figura 7 se muestra el histograma correspondiente a una imagen en escala de grises.

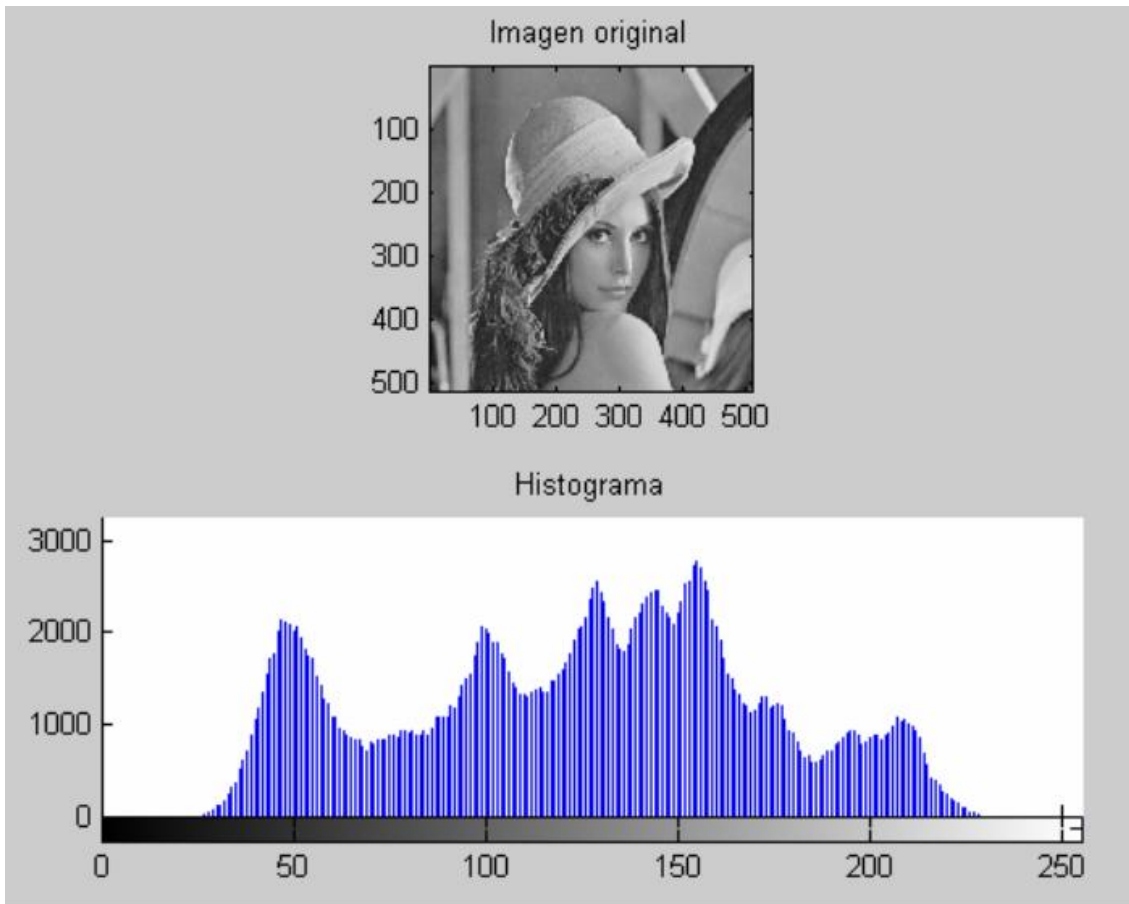


Figura 7. Histograma de una imagen en escala de grises

En la Figura 8 se muestra los histogramas correspondientes a cada uno de los canales RGB de una imagen en color.



Figura 8. Histogramas de los canales RGB de una imagen en color

### 2.1.5. Binarización de una imagen en escala de grises

La binarización de una imagen digital consiste en transformar la imagen en escala de grises en una imagen en blanco y negro. Para realizar la operación de binarización, se deberá elegir un valor adecuado de umbral dentro de los niveles de grises. Una vez elegido el umbral, todos los niveles de grises menores que el valor de umbral fijado se convertirán en negro y todos los que resulten ser mayores en blanco.

En la Figura 9 se puede ver el resultado de binarizar una imagen para distintos valores de umbral.



Figura 9. Binarización de una imagen según distintos niveles de umbral

### 2.1.6. Segmentación

La segmentación es el proceso mediante el cual una imagen se descompone en regiones o elementos que pueden corresponder a objetos o formas de interés. El proceso de segmentación se encarga de evaluar si cada pixel de la imagen pertenece o no al objeto de interés. Esta técnica de procesamiento de imágenes idealmente genera una imagen binaria, donde los pixeles que pertenecen al objeto se representan con un 1, mientras que los que no pertenecen al mismo se representan con un 0. Este tipo de particionamiento está basado en el análisis de alguna característica de la imagen, tal como los niveles de gris o la textura. A continuación se describe el método de segmentación basado en la umbralización.

#### 2.1.6.1. Segmentación basada en píxeles

Este método de segmentación toma en cuenta sólo el valor de gris de un pixel, para decidir si el mismo pertenece o no al objeto de interés. Para ello, se debe encontrar el rango de valores de gris que caracterizan dicho objeto, lo que requiere entonces la búsqueda y el análisis del histograma de la imagen.

El objetivo de este método, es el de encontrar de una manera óptima los valores característicos de la imagen que establecen la separación del objeto de interés, con respecto a las regiones que

no pertenecen al mismo; debido a esta característica y si los valores de gris del objeto y del resto de la imagen difieren claramente, entonces el histograma mostrará una distribución bimodal, con dos máximos distintos, lo que debiera generar, la existencia de una zona del histograma ubicada entre los dos máximos, que no presenten los valores característicos, y que idealmente fuera igual a cero, con lo cual se logrará una separación perfecta entre el objeto y la región de la imagen que lo circunda, al establecer un valor umbral ubicado en esta región del histograma. Al realizar la binarización de la imagen con dicho valor umbral obtendremos una imagen con el objeto de interés en blanco y el entorno en negro, o viceversa.

Si el valor del histograma ubicado entre los dos máximos, es distinto de cero, las funciones de probabilidad de los valores de gris del objeto y de la región restante, se solaparán, de tal manera que algunos píxeles del objeto deberán ser tomados como pertenecientes a la región circundante y viceversa. Conocida la distribución de la función de probabilidad de los píxeles del objeto y de la región circundante, es posible aplicar análisis estadístico en el proceso de buscar un umbral óptimo, con el número mínimo de correspondencias erróneas. Estas distribuciones pueden ser estimadas por histogramas locales, los cuales solamente incluyen las regiones correspondientes de la imagen.

Para la determinación del umbral óptimo, existen diferentes métodos. De entre ellos, el método de Otsu (1979) es uno de los más clásicos y populares. Otro método posible es la binarización mediante la media estadística del histograma como valor de umbral.

## **2.1.7. Métodos de binarización**

### **2.1.7.1. Binarización mediante el método de Otsu**

El método de Otsu resulta ser un método de binarización de imágenes digitales ampliamente utilizado en la literatura. Se basa en conceptos estadísticos, en concreto la dispersión de valores (en este caso, niveles de gris) a través de la varianza. Si partimos de una imagen con L niveles de intensidad, en nuestro caso 256, y asumiendo que el umbral buscado es T, las probabilidades hasta T y desde T hasta L resultan ser:

$$w1(t) = \sum_{z=1}^T P(z)$$

$$w2(t) = \sum_{z=T+1}^L P(z)$$

Donde ( $z$ ) es el total de apariciones del nivel de intensidad  $z$ . A continuación, se obtienen las medias y las varianzas asociadas:

$$\sigma_1(t)^2 = \sum_{z=0}^T (z - \mu_1(t))^2 \frac{P(z)}{w_1(t)}$$

$$\sigma_2(t)^2 = \sum_{z=T+1}^{L-1} (z - \mu_2(t))^2 \frac{P(z)}{w_2(t)}$$

Finalmente, se obtiene la varianza ponderada:

$$\sigma_w^2(t) = w_1(t) \cdot \sigma_1(t)^2 + w_2(t) \cdot \sigma_2(t)^2$$

Se elige el umbral  $T$  correspondiente al nivel de intensidad que proporcione la mínima varianza ponderada.

### **2.1.7.2. Binarización mediante la media**

Por media se entiende el valor medio de los niveles de intensidad en la imagen. Se obtiene directamente a partir del histograma de la imagen, mediante la siguiente ecuación:

$$m = \sum_{i=0}^{L-1} z_i \cdot P(z_i)$$

En imágenes industriales tenemos generalmente un alto contraste lo cual permite aplicar segmentaciones muy simples como el mencionado, para distinguir dos clases de regiones en la imagen: región objeto y región fondo.

### **2.1.8. Filtro de tamaño**

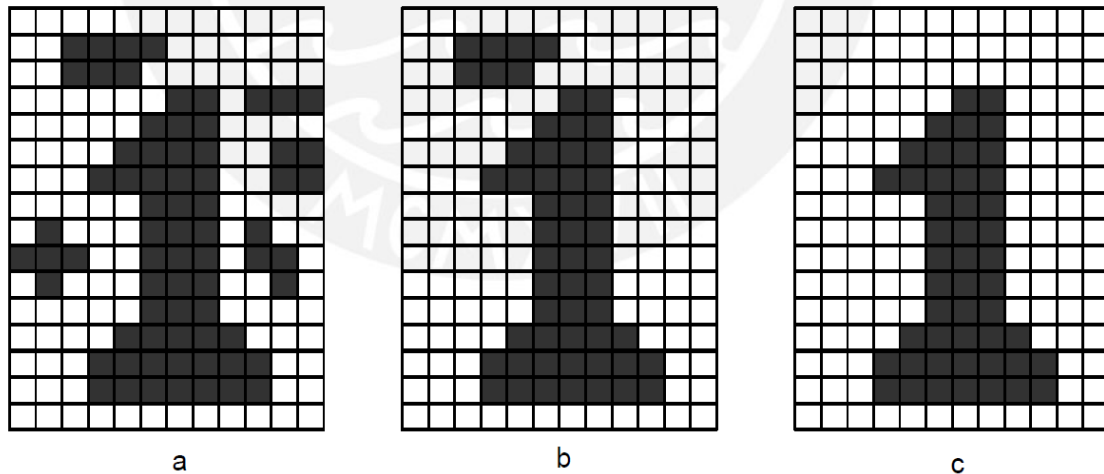
El filtrado de tamaño en imágenes binarias está orientado, principalmente con dos objetivos:

- Eliminación de ruido: debido al proceso de umbralización pueden aparecer regiones en la imagen binaria que son consecuencia del ruido. Tales regiones son normalmente pequeñas.
- Reducir el número de objetos de la imagen. En algunas ocasiones resulta interesante eliminar de una imagen binaria aquellos objetos que no superen un tamaño determinado, o por el contrario, eliminar los objetos que si superen este tamaño.



Los objetos son filtrados teniendo en cuenta su tamaño o área, o utilizando operaciones de erosión y dilatación.

Sin embargo, en muchas aplicaciones es conocido que los objetos de interés son de un tamaño mayor que  $T$  píxeles (área). Todas las componentes de la imagen que tienen un tamaño igual o menor que  $T$  píxeles se eliminan; para ello se cambian los correspondientes píxeles al valor del fondo (0).



**Figura 10. Eliminación de objetos de pequeño tamaño**

### **2.1.9. Etiquetado y Cuenta de objetos**

En la práctica llegar a la imagen binaria no suele ser suficiente para realizar una descripción adecuada, por lo que el proceso de segmentación se prolonga aplicando diversas técnicas sobre este tipo de imágenes.

Existe una gran cantidad de técnicas de análisis para imágenes binarias. Entre estas técnicas está el contar, etiquetar objetos y filtrado de objetos según su tamaño.

Una de las operaciones más comunes en visión es encontrar las componentes conectadas dentro de una imagen. En la Figura 11 vemos un ejemplo de una imagen y su imagen de componentes conectadas, en donde se ha etiquetado con un número a cada componente conectada u objeto

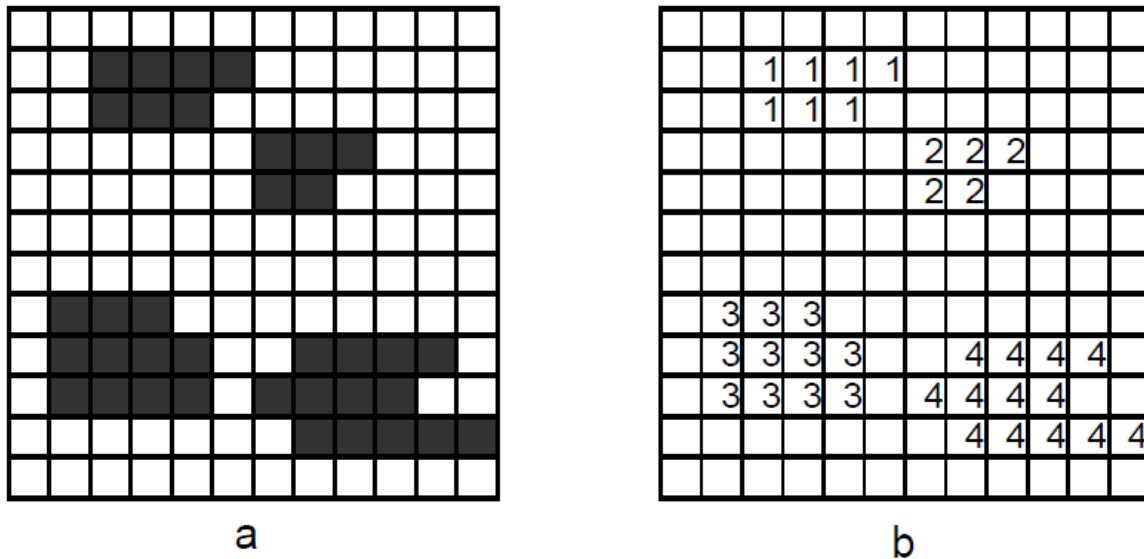


Figura 11. Etiquetado y cuenta de objetos

El etiquetado es una técnica que, partiendo de una imagen binaria, nos permite etiquetar cada uno de los objetos conectados presentes en la imagen. Esto va a posibilitar:

- Distinguir los objetos respecto del fondo (propiedad intrínseca de la imagen binaria).
- Distinguir un objeto respecto de los demás objetos.
- Conocer el número de objetos en la imagen.

## 2.2. Identificación y reconocimiento de caracteres

### 2.2.1. Descripción y extracción de características

#### 2.2.1.1. Introducción

Para reconocer un carácter u objeto es necesario tener una *descripción* de los mismos (descriptor o modelo del objeto). Los descriptores deben de ser independientes del tamaño, localización u orientación del objeto, y deben ser suficientes para discriminar objetos entre sí. Los descriptores se basan en la evaluación de alguna *característica* del objeto, por ejemplo:

- Descriptores unidimensionales* : códigos de cadena, perímetro, forma del perímetro.
- Descriptores bidimensionales* : área, momentos de inercia, etc.
- Descriptores específicos* : número de agujeros, área de agujeros, posición relativa de agujeros, rasgos diferenciadores de un objeto, etc.

Uno de los principales problemas en el reconocimiento de patrones, es encontrar una manera óptima de representar la información original que describe a cada uno de los patrones basado

en los *descriptores* mencionados inicialmente. Este problema es conocido como **extracción de características**. Este proceso de extracción de características trata de reducir la cantidad de información (reducción de dimensionalidad) que representa a cada uno de los patrones, obteniendo de esta forma, un **vector de características** que represente de la mejor manera posible al patrón original.

### **2.2.1.2. Extracción de características**

El término "*extracción de características*" se refiere al proceso de extraer algunas medidas numéricas e información *relevante* de los datos en bruto de los patrones suministrada por los sensores (representación inicial), para la clasificación. Por otro lado, se define también como el proceso de formar un conjunto de características más reducido que el correspondiente a la representación inicial.

Los vectores resultantes del proceso de **selección** o **extracción** se denominan **características** del objeto.

Conviene destacar las propiedades más importantes que deben barajarse en la elección o extracción de las características:

- a) **Discriminación:** valores numéricos diferentes para objetos de clases diferentes.
- b) **Fiabilidad:** cambios numéricos pequeños para objetos de una misma clase, es decir los objetos de una misma clase deberán representar la menor dispersión.
- c) **Incorrelación:** la independencia de las características equivale al principio de la parsimonia, decir lo mismo con la máxima economía de términos. Nunca debe utilizarse características que dependan fuertemente entre sí, ya que no añaden información. Se ha de tener la máxima información con el mínimo número de características.
- d) **Cálculo en Tiempo Real:** este es un requisito que puede llegar a ser determinante en ciertas aplicaciones de tiempo real, ya que las características deben calcularse en un tiempo aceptable.
- e) **Invarianza:** frente a transformaciones geométricas como rotación, traslación, escalado.
- f) **Dimensionalidad:** el tamaño del vector de características no debe de ser mayor que lo necesario. Las características deben representar una codificación óptima, se debe reflejar lo esencial del objeto, no lo accesorio.

### 2.2.1.3. Descriptores

Un descriptor podría definirse como un conjunto de números que describen una forma dada de un objeto en una imagen de entrada. Aunque la forma no sea completamente describible, el descriptor la permitirá discriminar de otras distintas.

De un descriptor se requiere que sea capaz de describir un mismo objeto en la imagen, independientemente de si este se encuentra en una posición distinta, con una rotación distinta, con un tamaño distinto (escalado) y con un punto de vista distinto (deformación). Se requiere por lo tanto que tenga la característica de invariante.

Para el cálculo de los descriptores se parte de una imagen previamente segmentada en regiones y/o donde las fronteras de los objetos han sido determinadas.

#### 2.2.1.3.1. Descriptores de forma simple

**Altura y ancho:** Número máximo de píxeles en la coordenada  $x$  y coordenada  $y$ .

**Área:** Número de píxeles que describen una forma, región segmentada o que se encuentran contenidos por una frontera, más número de píxeles de la frontera.

**Perímetro:** Número de píxeles que constituyen la frontera de una forma.

**Compactación:** La relación entre perímetro y área (es invariante a escala).

Está definida por  $p^2/A$ .

La forma más compacta es el círculo, el resto de formas tienen un valor de compactación mayor a la del círculo que es  $4 \cdot \pi$

**Elongación:** El ratio entre la altura y la anchura de la caja mínima que envuelve la frontera de una forma. Ayuda a conocer la forma de la región.

**Excentricidad o redondez:** Se mide como  $4 \cdot \pi \cdot A/p^2$ .

**Rectangularidad:** El cociente entre el área del objeto y el área de la caja que envuelve su frontera. Determina como de rectangular es una forma o región.

**Euler:** Describe propiedades de la forma (descriptor topológico). Se define como la diferencia entre número de componentes conectados y número de agujeros

### 2.2.1.3.2. Envoltentes

*Cierre convexo*: menor conjunto convexo que contiene la forma.

*Caja de Feret*: menor rectángulo, orientado según una referencia específica (p.ej. los ejes cartesianos), que encierra la forma.

*Menor rectángulo envolvente (MRE)*: menor rectángulo, orientado en cualquier dirección, que contiene la forma.

*Puntos extremo*: Los 8 puntos que definen una forma. Por ejemplo, superior-derecha, superior-izquierda, inferior-derecha, inferior-izquierda, derecha-superior, derecha-inferior, izquierda-superior e izquierda-inferior.

### 2.2.1.3.3. Distancias de signature o curvas

Son curvas que representan la evolución de distancias, como la distancia del centro de la región a sus píxeles de borde o frontera.

### 2.2.1.3.4. Momentos Generales

La teoría de los momentos proporciona una interesante y útil alternativa para la representación de formas de objetos. Si tenemos un objeto en una región que viene dado por los puntos en los que  $f(x,y) > 0$ , definimos el momento de orden  $p,q$  como:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p \cdot y^q \cdot I_0(x, y)$$

$$p, q = 0, 1 \dots \dots \infty$$

El interés de estos **momentos generales** radica en que dada una función acotada  $f(x,y)$  existe un conjunto de momentos generales y viceversa. Es decir, dado un conjunto de momentos generales se puede reconstruir una función  $f(x,y)$  única, simbólicamente:

$$f(x, y) \leftrightarrow m_{pq}$$

$$p, q = 0, 1 \dots \dots \infty$$

Esta correspondencia biunívoca no representaría ningún interés práctico, a menos que se pudiera reducir el número de momentos generales a una cantidad manejable.

Un momento de gran interés es el de orden cero (el orden de un momento viene dado por la suma de los índices p y q):

$$m_{00} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I_0(x, y)$$

que obviamente coincide con el *área del objeto* en píxeles.

Los momentos de orden uno,  $m_{01}$  y  $m_{10}$ , junto con  $m_{00}$  determinan el llamado *centro de gravedad del objeto*:

$$\bar{x} = \frac{m_{10}}{m_{00}} = \frac{\sum \sum x \cdot I(x, y)}{\sum \sum I(x, y)}$$

$$\bar{y} = \frac{m_{01}}{m_{00}} = \frac{\sum \sum y \cdot I(x, y)}{\sum \sum I(x, y)}$$

#### 2.2.1.3.5. Momentos Invariantes

El inconveniente de los momentos generales reside en el hecho de que varía con la posición del objeto dentro del plano de la imagen, así como con el tamaño relativo del objeto, que depende de la distancia entre la cámara y el objeto.

Los cambios en la posición del objeto se deben exclusivamente a las operaciones de giro o traslación. En cuanto a los cambios de tamaños se puede modelar como una operación de homotecia. Por tanto, se pueden transformar sucesivamente los momentos generales para hacerlos invariantes a traslaciones, giros y homotecias.

##### 2.2.1.3.5.1. Momentos invariantes a traslaciones

Los momentos generales se pueden hacer invariantes a traslaciones en el plano sin más que referirlos al centro de gravedad (x,y), obteniéndose los llamados *momentos centrales*:

$$u_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p \cdot (y - \bar{y})^q \cdot I_0(x, y)$$

Puesto que el centro de masas ocupa siempre la misma posición relativa respecto a todos los puntos del objeto, los momentos centrales no varían ante traslaciones de los objetos. El cálculo directo de los momentos centrales es posible en función de los momentos generales:

$$u_{pq} = \sum_{r=0}^p \sum_{s=0}^q \binom{p}{r} \binom{q}{s} (-\bar{x})^r \cdot (-\bar{y})^s \cdot m_{p-r,q-s}$$

Así aplicando esta expresión se obtendría los siguientes momentos centrales:

$$u_{00} = m_{00}$$

$$u_{01} = u_{10} = 0$$

$$u_{11} = m_{11} - \frac{m_{10} \cdot m_{01}}{m_{00}}$$

$$u_{20} = m_{20} - \frac{m_{10}^2}{m_{00}}$$

$$u_{02} = m_{02} - \frac{m_{01}^2}{m_{00}}$$

$$u_{03} = m_{03} - \frac{3 \cdot m_{01} \cdot m_{02}}{m_{00}} + \frac{2 \cdot m_{01}^3}{m_{00}^2}$$

$$u_{30} = m_{30} - \frac{3 \cdot m_{10} \cdot m_{20}}{m_{00}} + \frac{2 \cdot m_{10}^3}{m_{00}^2}$$

$$u_{21} = m_{21} - \frac{m_{01} \cdot m_{20}}{m_{00}} - \frac{2 \cdot m_{10} \cdot m_{11}}{m_{00}} + \frac{2 \cdot m_{10}^2 \cdot m_{01}}{m_{00}^2}$$

$$u_{12} = m_{12} - \frac{m_{10} \cdot m_{02}}{m_{00}} - \frac{2 \cdot m_{01} \cdot m_{11}}{m_{00}} + \frac{2 \cdot m_{01}^2 \cdot m_{10}}{m_{00}^2}$$

### 2.2.1.3.5.2. Momentos Invariantes a Homotecias

Suele emplearse la siguiente normalización, en donde los momentos centrales normalizados de orden  $p+q$  se define como:

$$n_{pq} = \frac{u_{pq}}{u_{00}^{\frac{p+q}{2}}}$$

donde

$$\gamma = \frac{p + q + 2}{2}$$

para  $p, q = 2, 3 \dots \dots \infty$

Estos momentos son además invariantes a homotecias.

### 2.2.1.3.5.3. Momentos invariantes a Traslaciones, Rotaciones y Homotecias

De los momentos de segundo y tercer orden, pueden derivarse 7 de los llamados "**momentos invariantes**" que, no dependen del tamaño ni la posición del objeto, pudiendo ser usados para **la identificación de objetos**. El siguiente conjunto de momentos invariantes se puede obtener usando únicamente los momentos centrales normalizados de orden 2 y 3:

$$\Phi_1 = n_{20} + n_{02}$$

$$\Phi_2 = (n_{20} - n_{02})^2 + 4 \cdot n_{11}^2$$

$$\Phi_3 = (n_{30} - 3 \cdot n_{12})^2 + (3 \cdot n_{21} - n_{03})^2$$

$$\Phi_4 = (n_{30} - 3 \cdot n_{12})^2 + (n_{21} + n_{03})^2$$

$$\begin{aligned} \Phi_5 = & (n_{30} - 3 \cdot n_{12})(n_{30} + n_{12})[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2] \\ & + (3 \cdot n_{21} - n_{03})(n_{21} + n_{03})[3 \cdot (n_{30} + n_{12})^2 - (n_{21} + n_{03})^2] \end{aligned}$$

$$\Phi_6 = (n_{20} - n_{02})[(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2] + 4 \cdot n_{11}(n_{30} + n_{12})(n_{21} + n_{03})$$

$$\begin{aligned} \Phi_7 = & (3 \cdot n_{21} - n_{03})(n_{30} + n_{12})[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2] \\ & + (3 \cdot n_{12} - n_{30})(n_{21} + n_{03})[3 \cdot (n_{30} + n_{12})^2 - (n_{21} + n_{03})^2] \end{aligned}$$

Este conjunto de momentos resulta ser **invariante a la traslación, la rotación y el cambio de escala**. Otra ventaja de los momentos es su gran facilidad de cálculo, sobre todo aplicándolos a imágenes binarias, ya que pueden ser calculados ópticamente a velocidades elevadas.



## **2.2.2. Reconocimiento de caracteres**

Una vez se tienen las características más importantes de la imagen a analizar se puede determinar el carácter correspondiente por medio de alguna de las técnicas de tratamiento de datos que se explican a continuación.

### **2.2.2.1. K-NN**

Para OCR, existe un método que proporciona muy buenos resultados. El algoritmo K-NN (K vecinos más próximos). Este método es muy popular debido a su sencillez y al buen comportamiento que presenta para afrontar diversos tipos de problemas de clasificación, siendo uno de ellos el de OCR.

Dado un conjunto de objetos prototipo de los que ya se conoce su clase (es decir, dado un conjunto de caracteres de muestra) y dado un nuevo objeto cuya clase no conocemos (imagen de un carácter a reconocer) se busca entre el conjunto de prototipos los “k” más parecidos a nuestro objeto. A este se le asigna la clase más numerosa entre los “k” objetos prototipo seleccionados.

#### ***Fase de entrenamiento y fase de test.***

Conociendo el funcionamiento básico del método de clasificación de los “k vecinos más próximos”, para poder empezar a trabajar con él es necesario reunir un conjunto de datos etiquetados, es decir, un conjunto de muestras prototipo con las clases a las que pertenecen.

En OCR, esta recolección implica disponer de una base de datos de imágenes de los tipos de caracteres que posteriormente se esperen reconocer. A este conjunto de datos se le denomina conjunto de entrenamiento. Sin embargo, la fase de entrenamiento no solo consiste en la recopilación de estos datos, sino que, típicamente, los datos originales que se dedican al entrenamiento deben ser preprocesados adecuadamente para obtener representaciones compactas y coherentes. En el ejemplo de OCR, esto quiere decir que las imágenes deben ser segmentadas (eliminación de ruido y selección de la caja mínima de inclusión), normalizadas y transformadas (extracción de características) para obtener los vectores que finalmente se almacenan como conjunto de entrenamiento.

Con este conjunto de entrenamiento ya construido, el clasificador K-NN ya puede ser utilizado para reconocer la clase de una nueva muestra. Esta es la fase de test y lógicamente, también aquí es necesario aplicar todo el preproceso descrito anteriormente a cada una de las nuevas muestras. Por lo tanto, aquí se ve la necesidad de disponer de métodos rápidos de realizar estas

tareas de preproceso, puesto que la velocidad de reconocimiento dependerá, en parte, de ellos. En la práctica se tiene que este preproceso es posible realizarlo muy rápidamente, aunque justo a continuación aparece la parte del proceso de reconocimiento que normalmente más carga computacional conlleva, la clasificación. De hecho es la parte que hace que el sistema sea computacionalmente más lento que otros métodos.

#### **2.2.2.2. Árboles de decisión**

Los árboles de decisión, al igual que el K-NN, es una técnica de tratamiento de datos que se puede aplicar en el contexto del reconocimiento óptico de caracteres. Los atributos que se quieren evaluar de un carácter determinado constituyen los nodos del árbol, mientras que los resultados finales de los mismos se almacenarán en las hojas del mismo. Tras la construcción del árbol y dada la estructura del mismo, toda la evaluación de caracteres se puede tratar como una arquitectura IF-THEN-ELSE, por lo que si el número de parámetros a evaluar es suficientemente grande para tener capacidad expresiva pero suficientemente pequeño para ser eficiente computacionalmente, el árbol puede resultar un método más rápido que el algoritmo K-NN.

#### **2.2.2.3. Redes neuronales**

Las redes neuronales son esquemas de tratamiento de datos que intentan imitar la arquitectura del cerebro. Se componen de una serie de unidades básicas, llamadas neuronas, que básicamente reciben una entrada, la multiplican por unos pesos y presentan una salida con una función de ajuste, que depende de la suma de salidas de la etapa anterior.

En el caso que nos ocupa, las redes de neuronas son una muy buena alternativa para la etapa de reconocimiento de caracteres, ya que una vez se entrenan, si este entrenamiento ha sido adecuado, pueden usarse para reconocer los caracteres que recibe como imágenes.

### **3. Objetivos**

Se pretende realizar la detección y el reconocimiento de caracteres presentes en imágenes de productos industriales. Ello pasa por alcanzar los siguientes objetivos.

1. Desarrollar algoritmos en MATLAB que realicen la detección de caracteres que aparecen en imágenes de productos industriales, incluso en el caso de que estos aparezcan poco contrastados.
2. Desarrollar algoritmos en MATLAB que realicen el reconocimiento de los caracteres detectados anteriormente.

### **4. Metodología**

La metodología que se sigue en la realización del presente trabajo es la siguiente.

- a) Se hace una revisión de las técnicas de visión artificial existentes que podemos utilizar en el reconocimiento óptico de caracteres, seleccionando aquellas que se piensa que pueden dar buenos resultados.
- b) Se desarrollan algoritmos basados en las técnicas seleccionadas y se evalúan los resultados.
- c) En el caso de no obtener resultados satisfactorios, se desarrollan algoritmos propios que mejoren los resultados que se obtienen.

### **5. Descripción de la solución**

#### **5.1. Definición del algoritmo propio de detección**

##### **5.1.1. Descripción del algoritmo**

Para la detección de los caracteres existentes en una imagen, y para poder conseguir una imagen binarizada donde los caracteres aparezcan en blanco y el resto de la imagen en negro, se ha explorado con el método de Otsu, porque de él se podía esperar que al menos en ciertas condiciones se pudieran obtener buenos resultados.

Como podremos ver en la descripción de las pruebas realizadas, dicho método es capaz de separar los caracteres de su entorno en ciertas condiciones particulares, pero en otros casos no es capaz de hacer esa separación de una manera satisfactoria.

Con la intención de mejorar el proceso de detección de caracteres en una imagen, se ha desarrollado un algoritmo con el que, dada una imagen convertida a escala de grises, se van haciendo agrupaciones de píxeles que tienen en común presentar niveles de gris cercanos, y se van seleccionando las agrupaciones que por sus características son susceptibles de corresponder a un carácter.

Para realizar la selección de dichas agrupaciones de píxeles u objetos, y teniendo en cuenta que los niveles de gris de los píxeles correspondientes a un carácter no han de ser muy diferentes, se procede a hacer un barrido sucesivo del histograma, escogiendo en cada momento los píxeles de la imagen con niveles de gris dentro de un rango. En la Figura 12 se puede ver un ejemplo de histograma correspondiente a una imagen en escala de grises, y los sucesivos rangos de niveles de gris. En dicho ejemplo, los rangos que se escogen son 0-50, 25-75, 50-100, 75-125, 100-150, 125-175, 150-200, 175-225 y 200-255. Se observa que se solapan, para facilitar que todos los objetos aparezcan, en un momento u otro, completos.

Para cada rango de niveles de gris, se almacenan los objetos susceptibles de ser carácter, y una vez recorridos todos los rangos, se obtiene como resultado una imagen binarizada en la que están presentes todos los objetos que han sido, en cualquiera de los rangos, identificados como un carácter.

Sobre dicha imagen se realizará el posterior proceso de reconocimiento de caracteres.

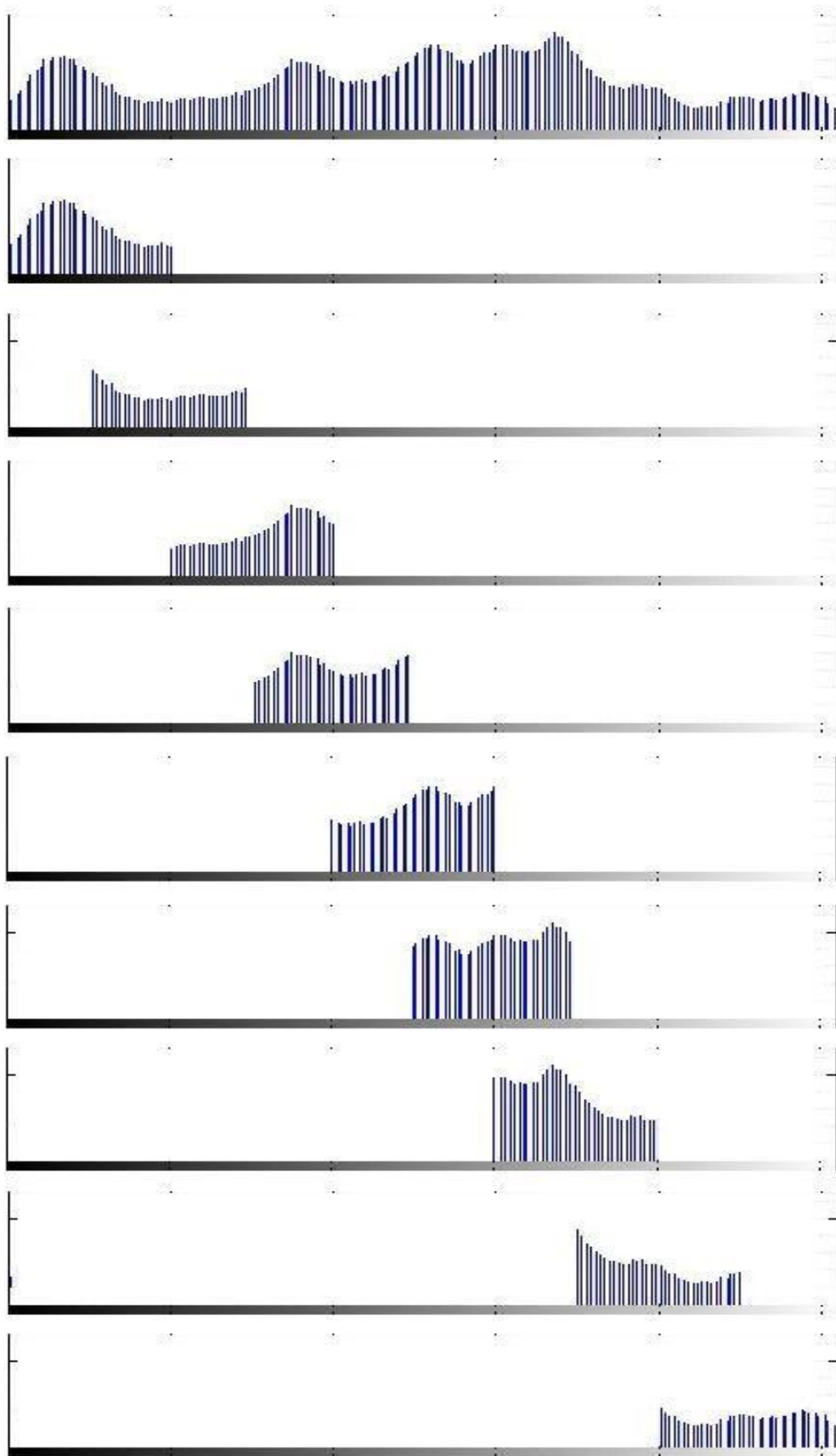


Figura 12. Fraccionamiento del histograma en diferentes rangos de niveles de gris

## 5.1.2. Pruebas realizadas

### 5.1.2.1. Pruebas con el método de Otsu

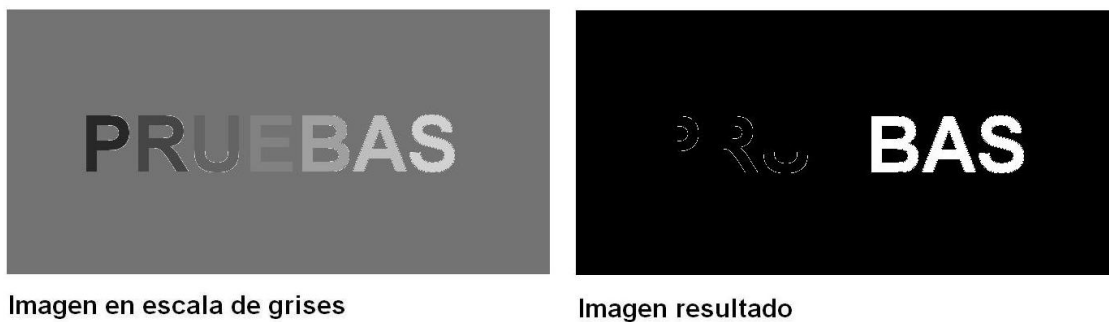
Se ha probado el método de Otsu en distintos casos, que podemos ver seguidamente.

En la Figura 13 podemos ver una imagen donde el nivel de gris de los caracteres es claramente distinto al nivel de gris del entorno. En este caso se ve que la aplicación del método de Otsu hace una separación de los caracteres respecto de su entorno.



**Figura 13. Aplicación método de Otsu en imagen con dos niveles de gris**

En la Figura 14 se da el caso de tener caracteres con distintos niveles de gris, con lo que la aplicación del método hace una separación insuficiente de los caracteres y de su entorno.



**Figura 14. Aplicación método de Otsu con varios niveles de gris en caracteres**

En la Figura 15 tenemos el caso de distintos niveles de gris tanto en los caracteres como en el entorno de los mismos. Al igual que en el caso anterior, también tenemos una separación insuficiente entre caracteres y resto de imagen.

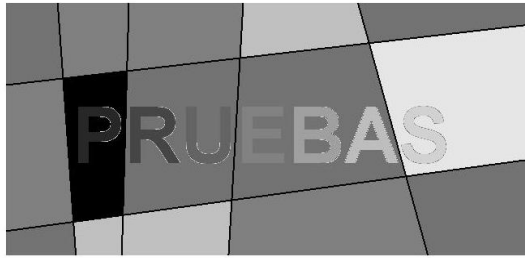


Imagen en escala de grises



Imagen resultado

**Figura 15. Aplicación método de Otsu con varios niveles de gris en caracteres y en entorno**

En la Figura 16 tenemos una imagen real de un producto industrial, en donde aparecen los caracteres identificativos de un neumático. Aunque se puede ver que los caracteres se distinguen visualmente del entorno por existir unas fronteras claras, el método de Otsu no da una separación clara en la imagen binarizada resultante.



Imagen en escala de grises

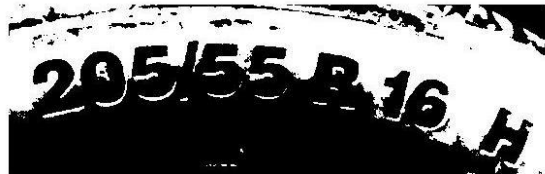


Imagen resultado

**Figura 16. Aplicación método de Otsu en imagen de producto industrial**

Sin embargo, en este otro caso (Figura 17) sí que se puede conseguir una separación de los caracteres respecto de su entorno más inmediato.



Imagen en escala de grises



Imagen resultado

**Figura 17. Aplicación método de Otsu en imagen de producto industrial**

Vemos que la aplicación del método de Otsu da buenos resultados cuando existen dos zonas claramente diferenciadas en el histograma de la imagen, pero cuando los niveles de gris son más dispersos no se obtienen buenos resultados.

### **5.1.2.2. Pruebas realizadas con el algoritmo desarrollado**

Se ha hecho una prueba del algoritmo desarrollado para cada una de las imágenes que anteriormente han sido ensayadas con el método de Otsu. En el caso de las dos imágenes de productos industriales, se puede ver que el resultado obtenido es casi completo. Se hace una separación completa de los caracteres respecto de su entorno, y aunque aparecen huecos dentro del área correspondiente a cada carácter, con un tratamiento posterior se pueden rellenar esos huecos y dejar la imagen lista para el proceso de reconocimiento.

En el caso de las imágenes de pruebas también se obtienen resultados claramente satisfactorios.

Vemos que en algunos de los casos se “cuelan” objetos que por sus características (forma, tamaño y otras propiedades) cumplen con los requisitos para ser considerados carácter, aún sin serlo.

En cualquier algoritmo de detección de caracteres, y particularmente también en el algoritmo desarrollado, se ha de tener en cuenta el tamaño esperado de los caracteres que van a aparecer en la imagen. Para el caso de las imágenes ensayadas el algoritmo ha tenido que ser modificado en la parte que depende del tamaño esperado de los caracteres, siendo distinto en el caso de cada una de las imágenes de productos industriales, y en el caso de las imágenes de pruebas( el mismo algoritmo para las tres, por tener el mismo tamaño de carácter). En total, tres algoritmos distintos.

En el caso de las imágenes industriales, el ensayo se ha realizado con rangos de niveles de gris, de anchura 50. En las imágenes de pruebas, la anchura de rango ha sido 20. En todos los casos, y tal y como se explicaba en la descripción del método, los rangos se solapan para favorecer la detección de objetos completos.

En cada una de las Figuras 18 y 19 aparece la imagen original, las imágenes binarizadas donde aparecen en blanco todos los píxeles con niveles de gris dentro del rango indicado, y finalmente la imagen binarizada, resultado de todo el proceso, donde aparecen todos los objetos detectados en cualquiera de los rangos.

Las Figuras 20, 21 y 22 corresponden a una imagen de pruebas, pero en este caso al tener una anchura de rango menor, aparecen más figuras ya que hay más rangos distintos.

De la misma forma sucede con las Figuras 23, 24 y 25, y con las Figuras 26, 27 y 28.

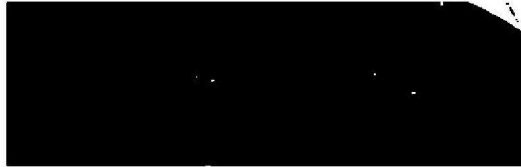




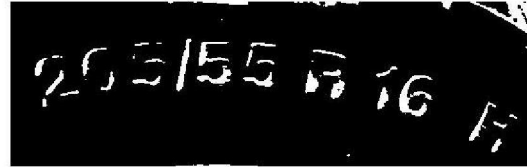
Imagen en escala de grises



Imagen con aumento de contraste



Niveles de gris en el rango 0-50



Niveles de gris en el rango 25-75



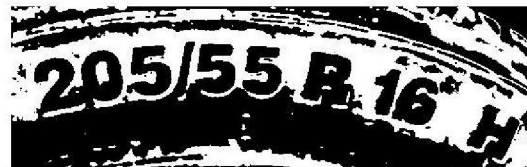
Niveles de gris en el rango 50-100



Niveles de gris en el rango 75-125



Niveles de gris en el rango 100-150



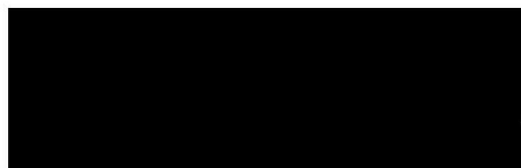
Niveles de gris en el rango 125-175



Niveles de gris en el rango 150-200



Niveles de gris en el rango 175-225



Niveles de gris en el rango 200-255



Imagen resultado

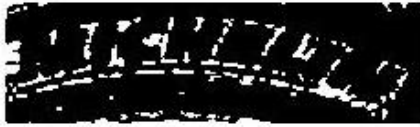
Figura 18. Algoritmo propio aplicado a imagen de producto industrial



Imagen en escala de grises



Imagen con aumento de contraste



Niveles de gris en el rango 0-50



Niveles de gris en el rango 25-75



Niveles de gris en el rango 50-100



Niveles de gris en el rango 75-125



Niveles de gris en el rango 100-150



Niveles de gris en el rango 125-175



Niveles de gris en el rango 150-200



Niveles de gris en el rango 175-225



Niveles de gris en el rango 200-255



Imagen resultado

Figura 19. Algoritmo propio aplicado a imagen de producto industrial

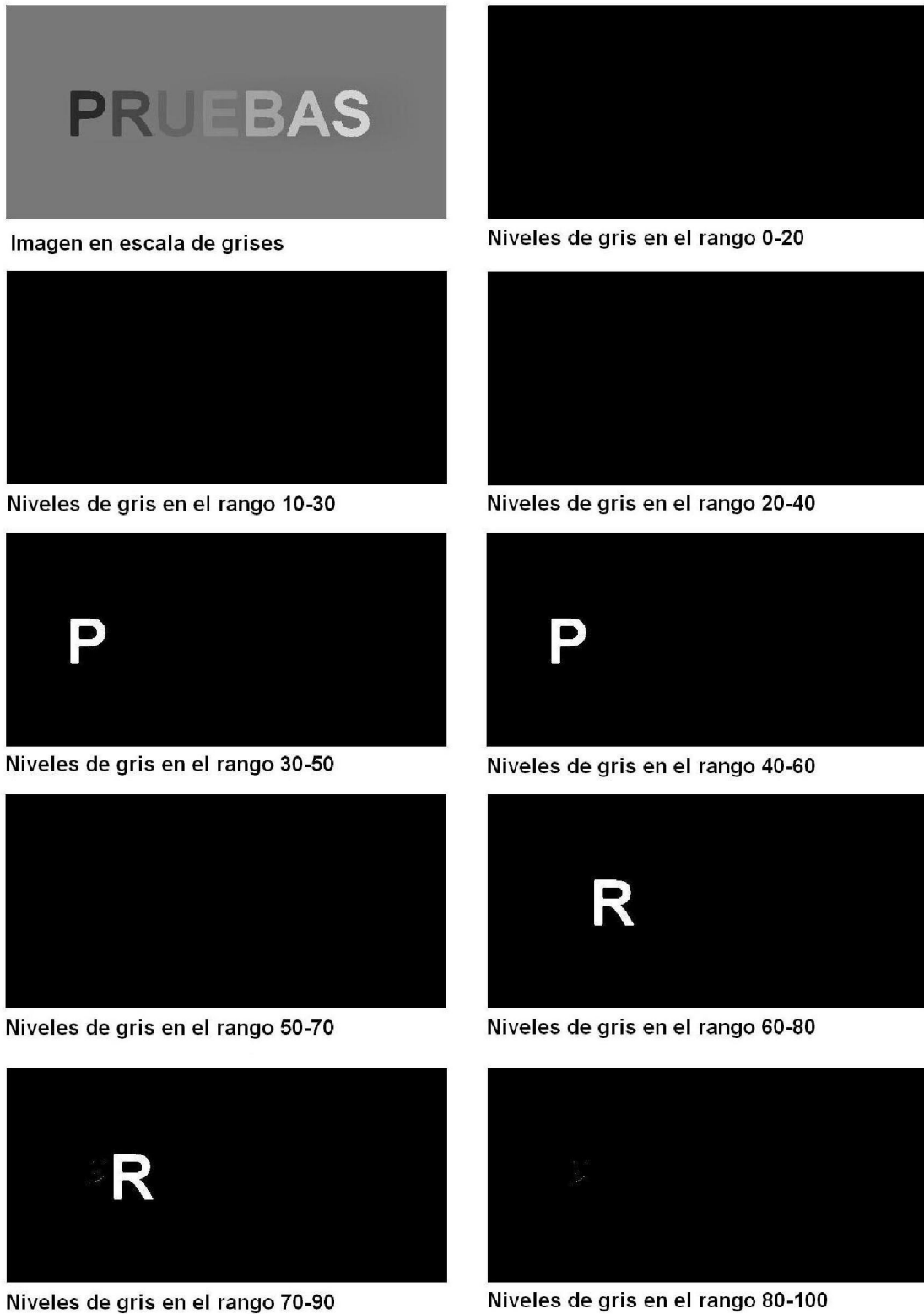
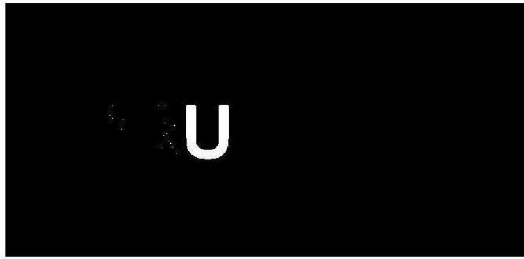


Figura 20. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (1)



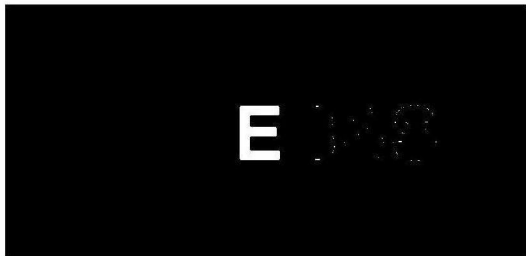
Niveles de gris en el rango 90-110



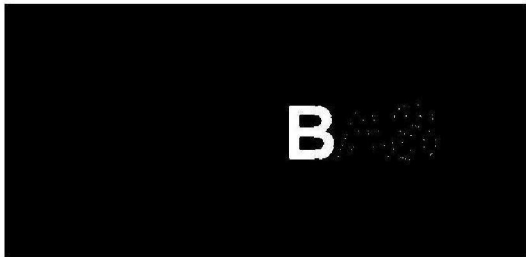
Niveles de gris en el rango 100-120

**PRUEBAS**

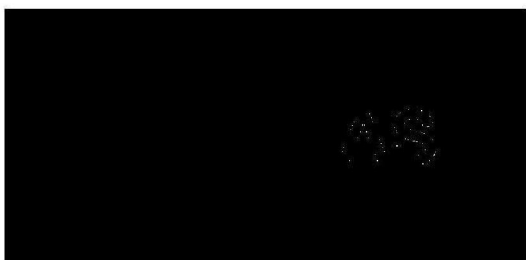
Niveles de gris en el rango 110-130



Niveles de gris en el rango 130-150



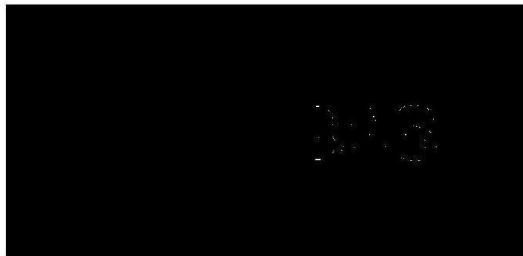
Niveles de gris en el rango 150-170



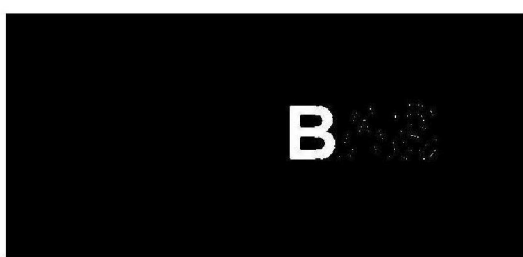
Niveles de gris en el rango 170-190

**PRUEBAS**

Niveles de gris en el rango 120-140



Niveles de gris en el rango 140-160



Niveles de gris en el rango 160-180



Niveles de gris en el rango 180-200

Figura 21. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (2)



Figura 22. Aplicación a imagen de pruebas fondo homogéneo-caracteres heterogéneos (3)



Imagen en escala de grises



Niveles de gris en el rango 0-20



Niveles de gris en el rango 10-30



Niveles de gris en el rango 20-40



Niveles de gris en el rango 30-50



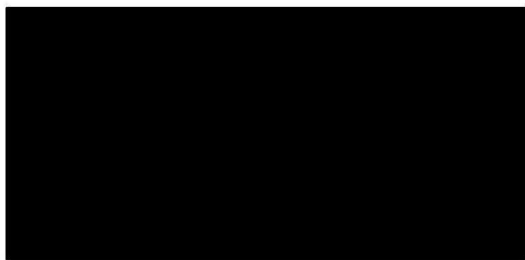
Niveles de gris en el rango 40-60



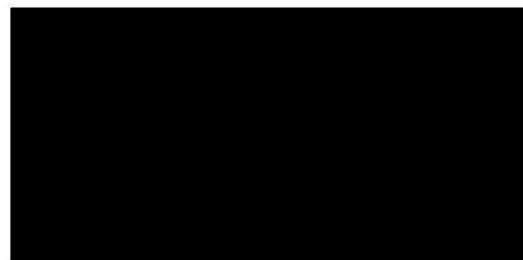
Niveles de gris en el rango 50-70



Niveles de gris en el rango 60-80



Niveles de gris en el rango 70-90



Niveles de gris en el rango 80-100

Figura 23. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (1)



Niveles de gris en el rango 90-110



Niveles de gris en el rango 100-120



Niveles de gris en el rango 110-130



Niveles de gris en el rango 120-140

**PRUEBAS**

Niveles de gris en el rango 130-150



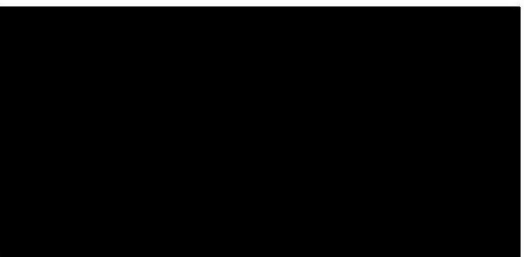
Niveles de gris en el rango 150-170

**PRUEBAS**

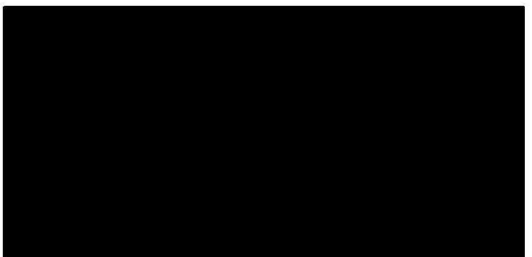
Niveles de gris en el rango 140-160



Niveles de gris en el rango 160-180



Niveles de gris en el rango 170-190



Niveles de gris en el rango 180-200

Figura 24. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (2)



Niveles de gris en el rango 190-210



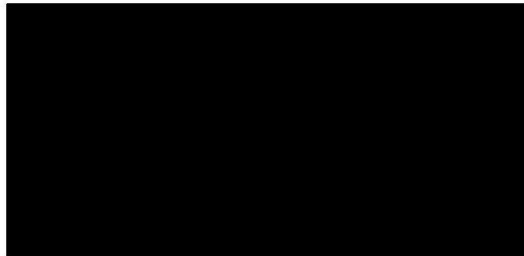
Niveles de gris en el rango 200-220



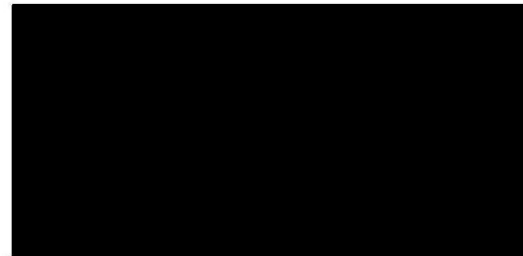
Niveles de gris en el rango 210-230



Niveles de gris en el rango 220-240



Niveles de gris en el rango 230-250



Niveles de gris en el rango 240-255



Imagen resultado

Figura 25. Aplicación a imagen de pruebas fondo homogéneo-caracteres homogéneos (3)



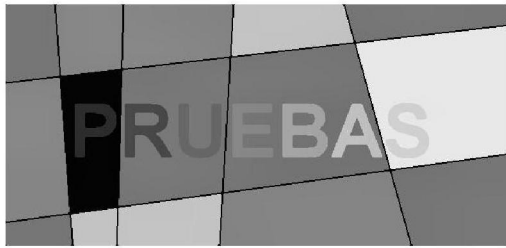
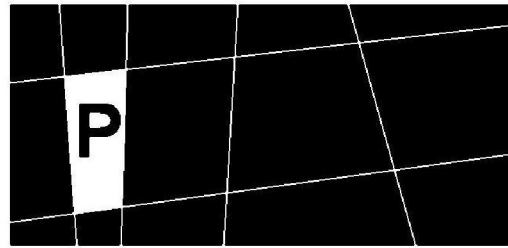
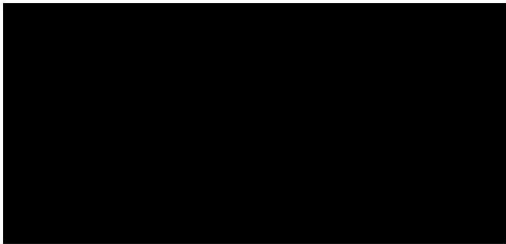


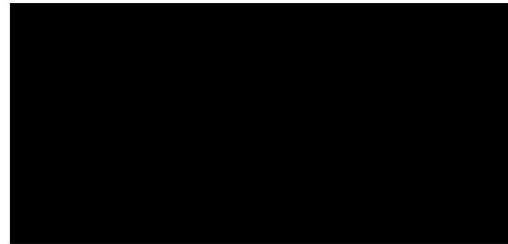
Imagen en escala de grises



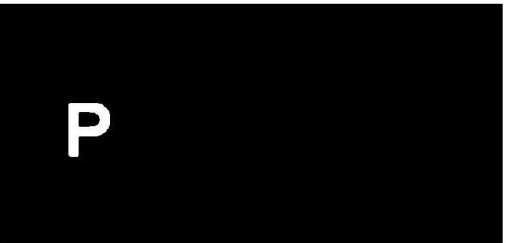
Niveles de gris en el rango 0-20



Niveles de gris en el rango 10-30



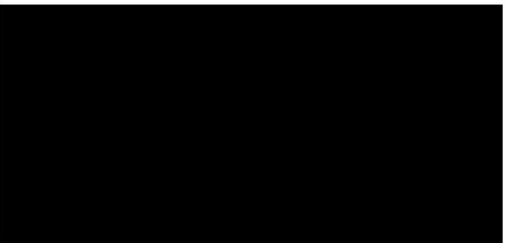
Niveles de gris en el rango 20-40



Niveles de gris en el rango 30-50



Niveles de gris en el rango 40-60



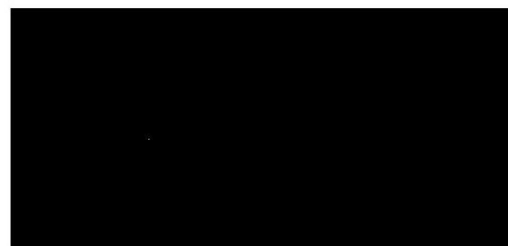
Niveles de gris en el rango 50-70



Niveles de gris en el rango 60-80

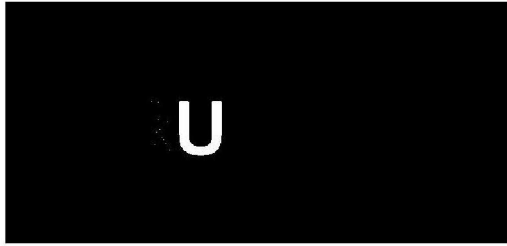


Niveles de gris en el rango 70-90



Niveles de gris en el rango 80-100

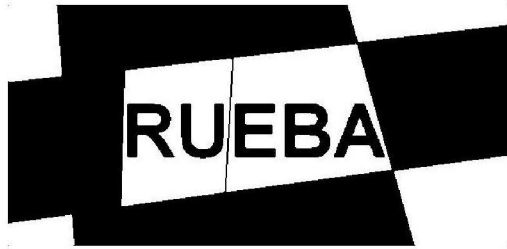
Figura 26. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (1)



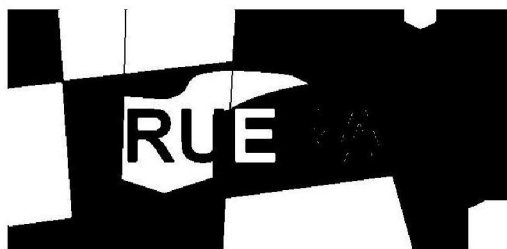
Niveles de gris en el rango 90-110



Niveles de gris en el rango 100-120



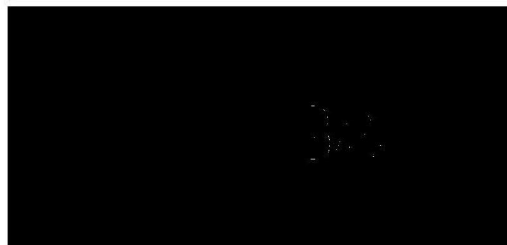
Niveles de gris en el rango 110-130



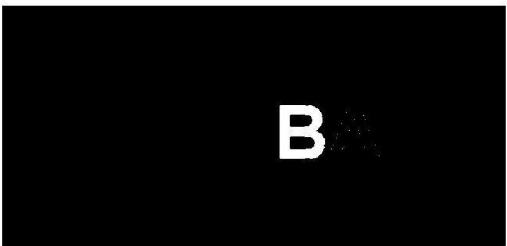
Niveles de gris en el rango 120-140



Niveles de gris en el rango 130-150



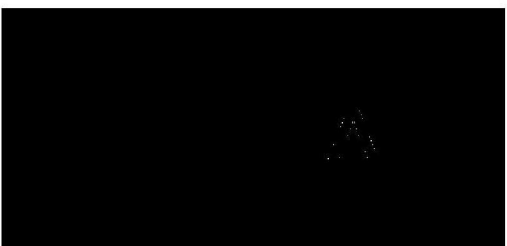
Niveles de gris en el rango 140-160



Niveles de gris en el rango 150-170



Niveles de gris en el rango 160-180



Niveles de gris en el rango 170-190



Niveles de gris en el rango 180-200

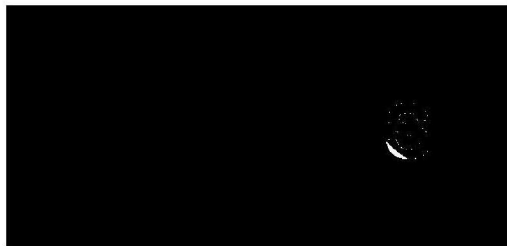
Figura 27. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (2)



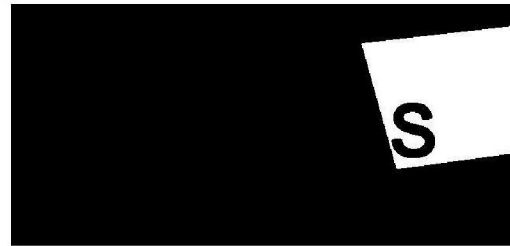
Niveles de gris en el rango 190-210



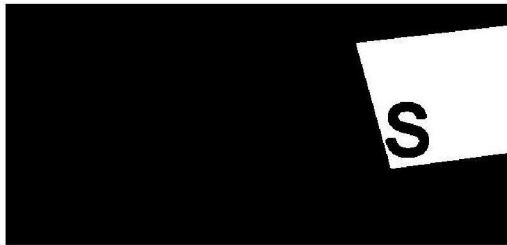
Niveles de gris en el rango 200-220



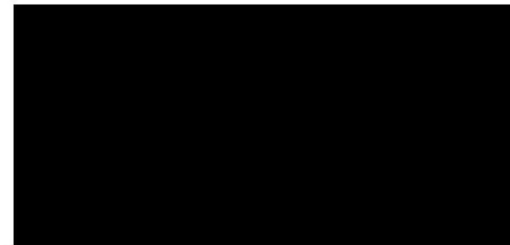
Niveles de gris en el rango 210-230



Niveles de gris en el rango 220-240



Niveles de gris en el rango 230-250



Niveles de gris en el rango 240-255



Imagen resultado

Figura 28. Aplicación a imagen de pruebas fondo heterogéneo-caracteres heterogéneos (3)

## 5.2. Definición del algoritmo propio de reconocimiento

### 5.2.1. Descripción del algoritmo

El algoritmo que se ha desarrollado para el reconocimiento de caracteres está basado en la técnica de reconocimiento K-NN.

Lo primero que se ha hecho ha sido crear una plantilla para cada carácter que queremos que sea reconocido. En dicha plantilla aparece el carácter con 6 tipos de fuente distintos, 4 tamaños distintos, y 4 estilos distintos (normal, negrita, *cursiva* y *negrita-cursiva*). Es decir,  $6 \times 4 \times 4 = 96$  letras distintas para cada carácter.

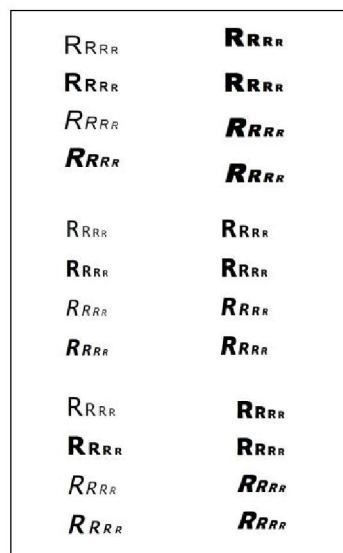


Figura 29. Plantillas de caracteres de muestra para generar matriz de entrenamiento

Una vez que se tienen las plantillas para cada uno de los caracteres, son leídas por un algoritmo que se encarga de crear una matriz de entrenamiento, donde cada fila corresponde a los datos de un carácter de los 96 que hay en cada plantilla. Por tanto, por cada letra o carácter habrá 96 conjuntos de valores, que corresponde a las 96 letras de cada plantilla, y que corresponde por tanto a 96 filas de la matriz de entrenamiento.

En cada fila de la matriz de entrenamiento hay 5 columnas, y en cada columna se almacenan los siguientes datos:

Columna 1: Letra

Columna 2: 7 momentos invariantes Hu.

Columna 3: Número de euler.

Columna 4: Número de endpoints en el carácter.

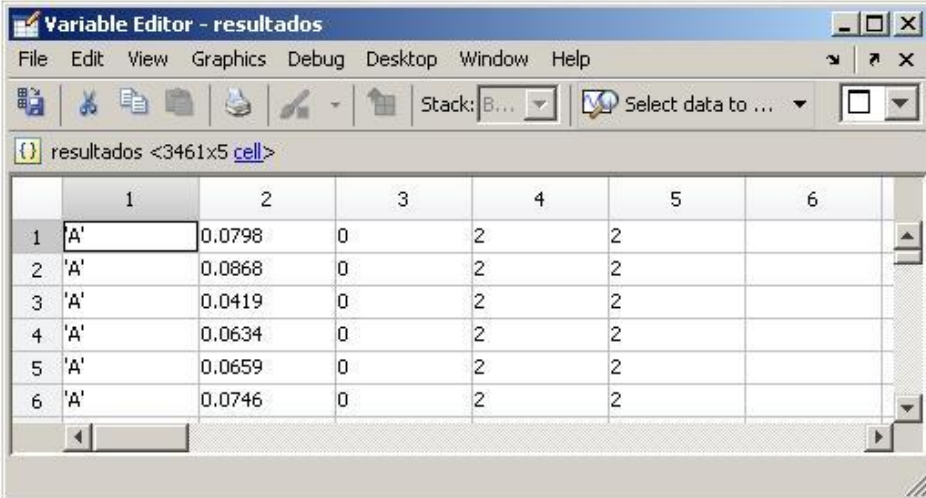
Columna 5: Número de branchpoints en el carácter.

	1	2	3	4	5	6
1	'A'	[0.3850 0.0130 0.0337 0.0013 3.6975e-06 1.2995e-04 7.6934e-06]	0	2	2	
2	'A'	[0.3880 0.0026 0.0417 0.0012 -8.4876e-06 6.0924e-05 -7.5024e-07]	0	2	2	
3	'A'	[0.2742 0.0012 0.0129 2.2602e-04 -3.8573e-07 7.6631e-06 2.9440e-09]	0	2	2	
4	'A'	[0.2502 0.0045 0.0088 4.0365e-04 5.5079e-07 2.3967e-05 5.2295e-07]	0	2	2	
5	'A'	[0.3719 0.0092 0.0284 5.0547e-04 -1.9161e-06 4.8426e-05 -3.9574e-08]	0	2	2	
6	'A'	[0.2388 0.0024 0.0064 6.9864e-05 -4.6832e-08 3.4229e-06 1.9496e-09]	0	2	2	
7	'A'	[0.3856 0.0195 0.0305 0.0014 7.3559e-06 1.0426e-04 4.6294e-06]	0	2	2	
8	'A'	[0.3126 0.0123 0.0150 7.9464e-04 1.7948e-06 6.5258e-05 2.0710e-06]	0	2	2	
9	'A'	[0.3745 0.0132 0.0302 8.7084e-04 2.2675e-06 8.0130e-05 3.8484e-06]	0	2	2	
10	'A'	[0.3785 0.0033 0.0383 9.3323e-04 -5.5676e-06 5.3117e-05 -4.0591e-07]	0	2	2	
11	'A'	[0.2709 0.0014 0.0128 2.0372e-04 -3.2719e-07 7.6419e-06 -3.1242e-08]	0	2	2	
12	'A'	[0.2455 0.0048 0.0083 3.5837e-04 4.7158e-07 2.0882e-05 4.0103e-07]	0	3	3	
13	'A'	[0.2812 0.0059 0.0108 3.5932e-04 4.2806e-07 2.2086e-05 5.6367e-07]	0	2	2	
14	'A'	[0.3898 0.0137 0.0334 0.0012 3.7512e-06 1.1192e-04 6.3769e-06]	0	2	2	
15	'A'	[0.3821 0.0063 0.0329 6.5087e-04 -2.9046e-06 5.0274e-05 7.9429e-07]	0	2	2	
16	'A'	[0.2737 0.0022 0.0104 1.8102e-04 -2.4096e-07 8.3586e-06 6.1872e-08]	0	2	2	
17	'A'	[0.3781 0.0084 0.0298 6.3504e-04 -2.7518e-06 5.7883e-05 2.6274e-07]	0	2	2	
18	'A'	[0.3966 0.0212 0.0319 0.0018 1.0621e-05 1.7661e-04 9.4427e-06]	0	2	2	
19	'A'	[0.2391 0.0026 0.0063 7.2268e-05 -4.8635e-08 3.6616e-06 3.2730e-09]	0	2	2	
20	'A'	[0.3141 0.0126 0.0149 8.8528e-04 1.8631e-06 7.6710e-05 2.6164e-06]	0	2	2	
21	'A'	[0.3636 0.0136 0.0271 6.6623e-04 1.9825e-06 5.3674e-05 2.0191e-06]	0	2	2	
22	'A'	[0.3700 0.0035 0.0345 7.5715e-04 -3.8716e-06 4.4285e-05 -2.0961e-08]	0	2	2	
23	'A'	[0.2713 0.0018 0.0103 2.1507e-04 3.2016e-07 9.2248e-06 -1.3849e-08]	0	2	2	
24	'A'	[0.2456 0.0052 0.0066 3.1103e-04 3.0096e-07 1.9218e-05 3.3038e-07]	0	2	2	
25	'A'	[0.3799 0.0136 0.0315 8.4542e-04 2.7539e-06 7.0770e-05 3.3807e-06]	0	2	2	
26	'A'	[0.3711 0.0067 0.0311 3.9465e-04 -1.3229e-06 3.1307e-05 3.9964e-07]	0	2	2	
27	'A'	[0.2833 0.0062 0.0111 3.7538e-04 4.9410e-07 2.2100e-05 5.8780e-07]	0	2	2	
28	'A'	[0.2771 0.0027 0.0109 1.7475e-04 -2.4034e-07 8.9615e-06 1.6869e-08]	0	2	2	

Figura 30. Matriz de entrenamiento

La descripción del proceso que sigue el algoritmo desarrollado es la siguiente.

- Recibe como dato de partida una imagen binarizada con los caracteres en blanco y el fondo en negro.
- Realiza el etiquetado y cuenta de objetos presentes en la imagen.
- Carga la matriz de entrenamiento respecto a la cual se van a comparar las propiedades de cada uno de los caracteres presentes en la imagen.
- Abre un fichero .txt donde va a ir escribiendo los caracteres que va a ir reconociendo.
- Para cada objeto o carácter presente en la imagen:
  - Obtiene una imagen con solo ese carácter.
  - Realiza el cálculo de los momentos invariantes Hu de dicho carácter.
  - Crea una matriz de resultados, fila a fila, donde mantiene las columnas y datos de la matriz de entrenamiento, a excepción de la columna de momentos Hu, que la sustituye por la distancia euclídea del vector de momentos Hu de nuestro carácter respecto al vector de momentos Hu de cada carácter de la matriz de entrenamiento.



	1	2	3	4	5	6
1	'A'	0.0798	0	2	2	
2	'A'	0.0868	0	2	2	
3	'A'	0.0419	0	2	2	
4	'A'	0.0634	0	2	2	
5	'A'	0.0659	0	2	2	
6	'A'	0.0746	0	2	2	

**Figura 31. Matriz de resultados**

- Ordena la matriz de resultados obteniendo una matriz de resultados ordenados según valores crecientes de distancia euclídea.
- Determina el número de euler, el número de endpoints y el número de branchpoints de nuestro carácter.
- Recorre la matriz de resultados ordenados de manera ascendente, hasta alcanzar la fila correspondiente a un carácter de muestra en dónde coincidan los números de euler, endpoints y branchpoints con los de nuestro carácter.
- Se asigna a nuestro carácter el valor letra correspondiente a esa fila de la matriz. Se escribe la letra en el fichero .txt.

- Se cierra el fichero de escritura, en donde se han ido escribiendo todos los caracteres reconocidos. Dicho fichero muestra como salida del algoritmo el resultado del reconocimiento de todos los caracteres presentes en la imagen.

Se han realizado dos variantes del algoritmo de reconocimiento de caracteres.

1. Reconocimiento de caracteres con la forma original de los mismos y con matriz de entrenamiento construida a partir de muestras originales.
2. Reconocimiento de caracteres tras un proceso de esqueletización de los mismos, y con matriz de entrenamiento construida a partir de muestras esqueletizadas.

### **5.2.2. Pruebas realizadas**

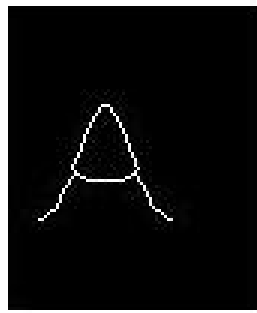
En el proceso de reconocimiento de caracteres descrito se hace la determinación de los puntos finales o endpoints y de los puntos de ramificación o branchpoints. Para ello, se realiza una esqueletización del carácter.

Pero en la obtención del esqueleto sucede a veces que donde hay realmente un endpoint aparecen dos, e igualmente en el caso de los branchpoints. Pese a ello, con un sencillo algoritmo se puede determinar el número real de endpoints y branchpoints.

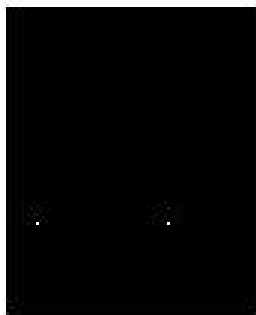
Para ello se recurre a dilatar tanto los endpoints como los branchpoints. Dos endpoints muy cercanos dilatados se convierten en un único área conectada, pero dos endpoints reales dilatados se convierten en dos áreas separadas. Igualmente sucede con los branchpoints.



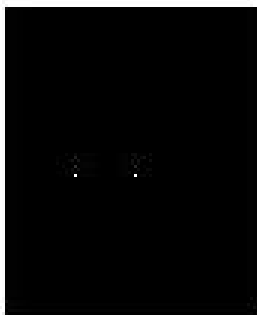
Caracter



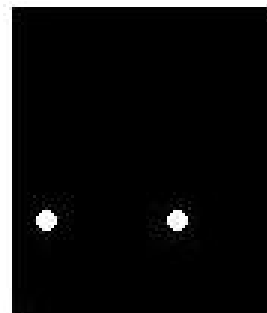
Caracter esqueletizado



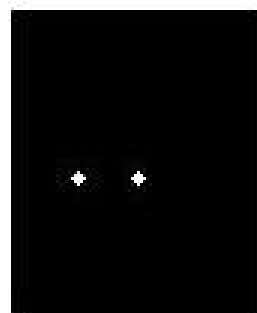
Endpoints



Branchpoints



Endpoints dilatados



Branchpoints dilatados

**Figura 32. Determinación de endpoints y branchpoints para la letra A**

Número de endpoints y branchpoints que aparecen en el esqueleto

$$nep=2$$

$$nbp=2$$

Número de endpoints y branchpoints dilatados

$$nep\_dil=2$$

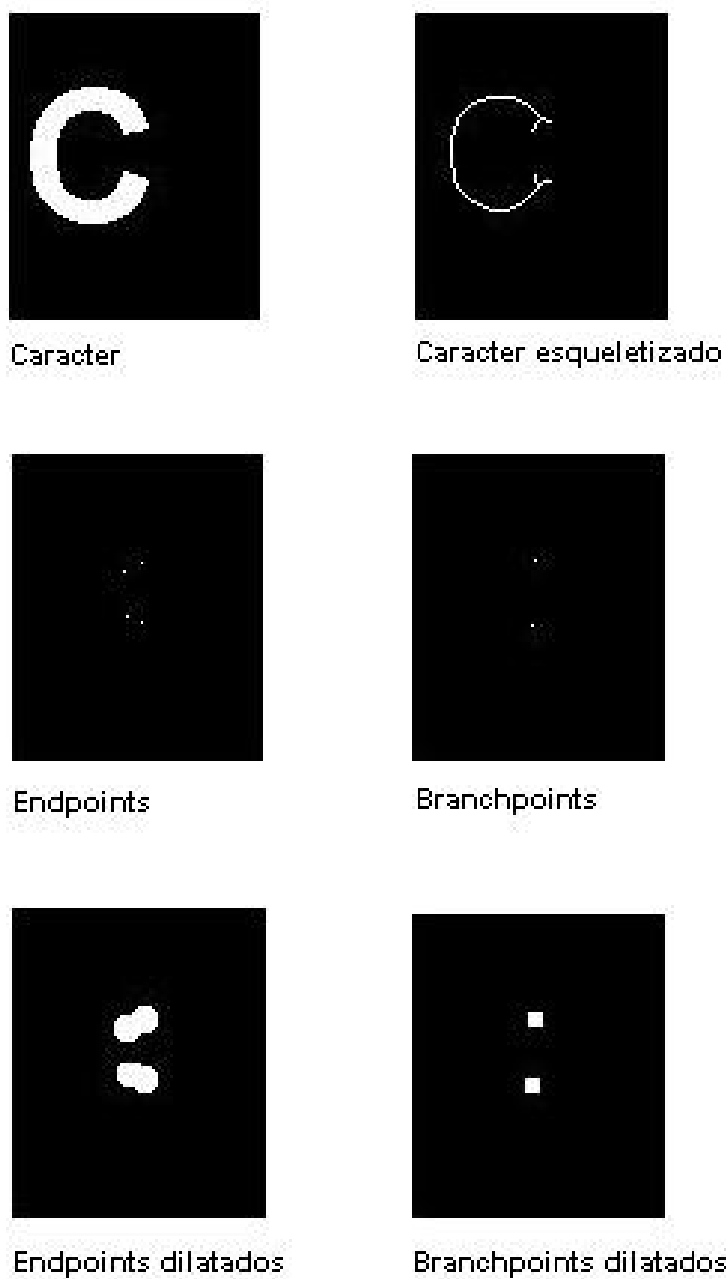
$$nbp\_dil=2$$

Número real de endpoints y branchpoints

$$nep\_real=nep\_dil=2$$

$$nbp\_real=nbp\_dil-(nep-nep\_dil)=2-(2-2)=2$$





**Figura 33. Determinación de endpoints y branchpoints para la letra C**

Número de endpoints y branchpoints que aparecen en el esqueleto

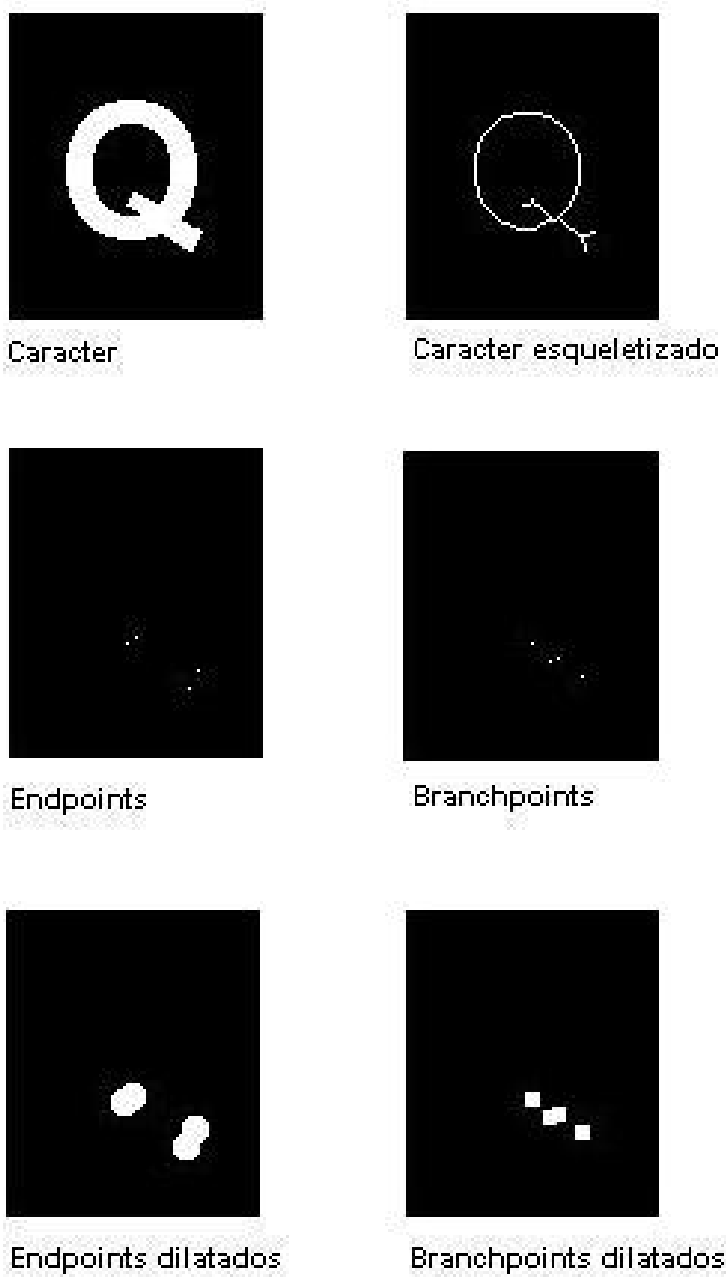
$$nep=4 \qquad \qquad \qquad nbp=2$$

Número de endpoints y branchpoints dilatados

$$nep_{dil}=2 \qquad \qquad \qquad nbp_{dil}=2$$

Número real de endpoints y branchpoints

$$nep_{real}=nep_{dil}=2 \qquad \qquad \qquad nbp_{real}=nbp_{dil}-(nep-nep_{dil})=2-(4-2)=0$$



**Figura 34. Determinación de endpoints y branchpoints para la letra Q**

Número de endpoints y branchpoints que aparecen en el esqueleto

$$nep=4$$

$$nbp=4$$

Número de endpoints y branchpoints dilatados

$$nep\_dil=2$$

$$nbp\_dil=3$$

Número real de endpoints y branchpoints

$$nep\_real=nep\_dil=2$$

$$nbp\_real=nbp\_dil-(nep-nep\_dil)=3-(4-2)=1$$

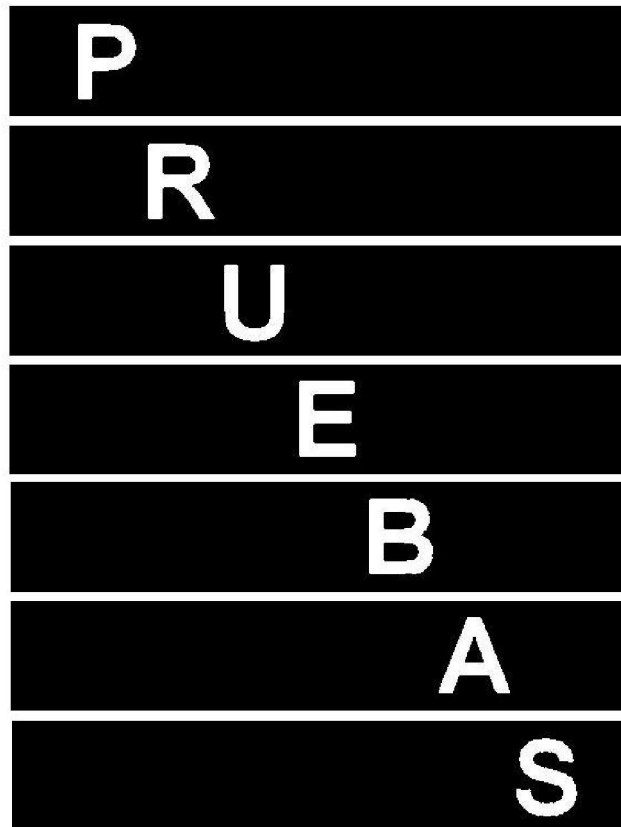
Los resultados obtenidos de la aplicación de los algoritmos de reconocimiento de caracteres son los mostrados en las siguientes figuras.

**Primer caso:** Reconocimiento de caracteres con forma original



PRUEBAS

Imagen binarizada



P  
R  
U  
E  
B  
A  
S

Objetos de reconocimiento



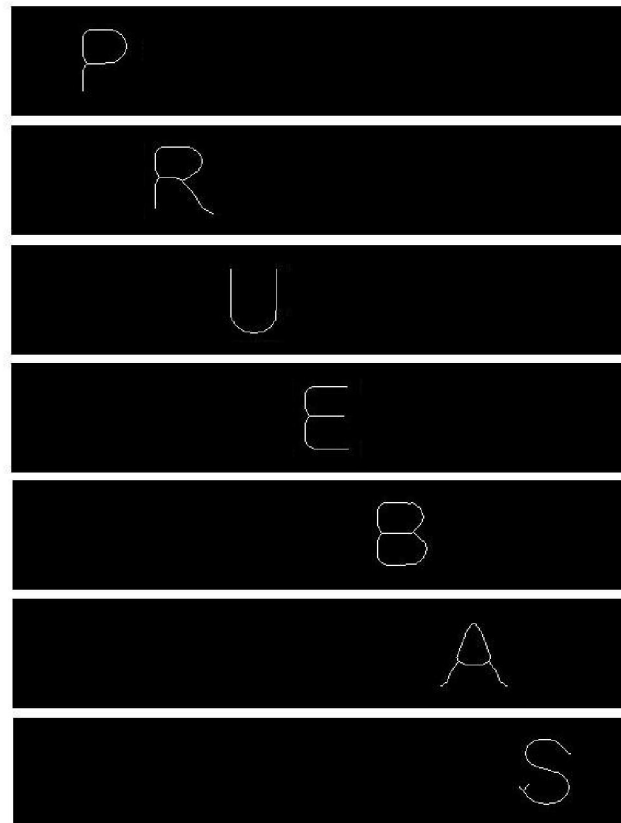
Resultado del proceso de reconocimiento

Figura 35. Reconocimiento de caracteres con forma original

Segundo caso: Reconocimiento de caracteres esqueletizados

**PRUEBAS**

Imagen binarizada



Objetos de reconocimiento



Resultado del proceso de reconocimiento

Figura 36. Reconocimiento de caracteres esqueletizados

Vemos que en el caso del procesamiento sobre caracteres esqueletizados se obtienen mejores resultados que sobre caracteres originales.

## **6. Conclusiones**

Antes de abordar la realización del trabajo se ha hecho una revisión de las técnicas de visión artificial existentes, que pueden ser utilizadas en la detección y reconocimiento de caracteres. Se han ensayado aquellas que mejor podían servir, y se ha visto las ventajas e inconvenientes que presentan a la hora de aplicarlas en nuestro caso.

La primera parte del trabajo ha sido la de detección de caracteres. En ella se han ensayado técnicas conocidas que permiten dicha detección, y aunque en algunos casos se han obtenido buenos resultados, no ha sido así en otros. Con la intención de mejorar la detección en estos casos, se ha desarrollado un algoritmo propio que permite detectar y separar los caracteres, incluso aquellos que aparecen con poco contraste respecto de su entorno. Se puede ver como en todas las pruebas que se han hecho se ha conseguido una separación de los caracteres, incluso en las imágenes de neumáticos, donde la variabilidad en los niveles de gris, al no ser una superficie plana, dificultaban la tarea.

En la segunda parte, la de reconocimiento de caracteres, se ha visto que la técnica K-NN (K Nearest Neighbours) podía dar buenos resultados, y se ha desarrollado un algoritmo de reconocimiento que se asemeja a dicha técnica. Se han hecho pruebas sobre las dos variantes del algoritmo, pudiendo comprobar que la variante que evalúa los caracteres esqueletizados arroja mejores resultados que la que evalúa los caracteres originales

Se han alcanzado resultados satisfactorios, al mismo tiempo que mejorables, lo que nos lleva a pensar que una continuación por esta línea de trabajo puede llegar a dar unos buenos resultados en cuanto a calidad y fiabilidad.

Se ha hecho por tanto una incursión en un campo de la visión artificial, el del reconocimiento de caracteres, sobre el que ya se han desarrollado técnicas que dan muy buenos resultados en multitud de aplicaciones reales, pero en el que al mismo tiempo queda mucho por investigar

## 7. Bibliografía y Referencias

Master Automática y Robótica. Universidad de Alicante  
Apuntes de clase de la asignatura Sistemas de Percepción.

Carlos Javier Sánchez y Víctor Sardonís. Universidad Carlos III. Madrid  
Reconocimiento óptico de caracteres (OCR).

Arturo de la Escalera, 2001. Visión por Computador: Fundamentos y Métodos,  
Ed. Prentice Hall.

Gonzalo Pajares, Jesús M. de la Cruz, 2001. Visión por computador: imágenes digitales y aplicaciones. RA-MA Editorial.

Otsu, N. (1979). A threshold selection method from gray-level histogram. IEEE Transactions on System Man and Cybernetics 9, 62-66, 1979.

Abdelmalik Moujahid, Iñaki Inza y Pedro Larrañaga, 2000. Clasificadores K-NN.  
Departamento de Ciencias de la Computación e Inteligencia Artificial Universidad del País Vasco.

OCR para caracteres impresos basados en árboles binarios.

<http://alojamientos.us.es/gtocom/pid/pid10/OCRRarbolbinario.htm>

José R. Hilera González, Juan P. Romero Villaverde, José

A. Gutiérrez de Mesa. Sistema de Reconocimiento Optico de Caracteres (OCR) con Redes Neuronales.

[http://www.cc.uah.es/hilera/docs/1996/c\\_jiacse1/c\\_jiacse1.htm](http://www.cc.uah.es/hilera/docs/1996/c_jiacse1/c_jiacse1.htm)

## 8. Anexos

### Anexo I: Código MATLAB

```
%Cuerpo principal del programa de detección y reconocimiento de caracteres  
%en la versión de evaluación de caracteres originales
```

```
I=imread('PRUEBAS.bmp');
```

```
%Llamada a la función de detección  
char_found=deteccion_caracteres(I);  
imwrite(char_found,'char_found.bmp');
```

```
%Llamada a la función de reconocimiento  
reconocimiento_caracteres(char_found);
```

```
%Cuerpo principal del programa de detección y reconocimiento de caracteres  
%en la versión de evaluación de caracteres esqueletizados
```

```
I=imread('PRUEBAS.bmp');
```

```
%Llamada a la función de detección  
char_found=deteccion_caracteres(I);  
imwrite(char_found,'char_found.bmp');
```

```
%Llamada a la función de reconocimiento  
reconocimiento_caracteres_thin(char_found);
```

```

function [ objetos_detectados ] = deteccion_caracteres( imagen )

% Esta función recibe una imagen y devuelve una imagen
% binarizada con los caracteres detectados en blanco

i1=imagen;           %Se carga la imagen
i2=rgb2gray(i1);     %Se convierte a escala de grises
[x,y]=size(i2);      %Se determina el tamaño n° filas y n°columnas
im1=medfilt2(i2,[3,3]); %Se aplica filtro de mediana
im2=adapthisteq(im1); %Se realiza un aumento del contraste
objeto_F=zeros(x,y); %Se pone en negro la imagen que va a almacenar todos los objetos
                    %que se van a detectar e identificar como un carácter

%Realiza un barrido sucesivo por el histograma
for z=0:25

    B=bin_umbrales_2(im2,z); %Devuelve sucesivamente una imagen con píxeles con un nivel de gris
                             %dentro de un rango en blanco y los demás en negro
    C=im2bw(B,128/256);      %Se binariza la imagen
    D=bwareaopen(C,100);     %Se eliminan pequeños objetos con tamaño <100 píxeles
    [region numr]=bwlabel(C,8); %Se realiza el etiquetado y cuenta de objetos

    %Determina para cada objeto etiquetado si es un carácter
    for etiqueta=1:numr
        objeto=zeros(x,y); % Se pone en negro la imagen que va a almacenar cada objeto etiquetado

        %La matriz objeto almacena el objeto etiquetado con el valor de etiqueta
        for i=1:x
            for j=1:y
                if region(i,j)==etiqueta
                    objeto(i,j)=C(i,j);
                else
                    objeto(i,j)=objeto(i,j);
                end
            end
        end

        props=regionprops(objeto,'all'); %Se extraen las propiedades del objeto

        %Se asignan a distintas variables los valores de distintos descriptores
        area=props.Area;
        extent=props.Extent;
        solidez=props.Solidity;
        ED=props.EquivDiameter;
        FA=props.FilledArea;
        MAL=props.MajorAxisLength;
        minAL=props.MinorAxisLength;
        perimeter=props.Perimeter;
        ex=props.Extrema;

        %Detecta si el objeto está en el borde de la imagen
        borde=0;
        for k=1:8
            if((ex(k,1)<1)||((ex(k,1)>(y-1))||(ex(k,2)<1)||((ex(k,2)>(x-1))))
                borde=1;
            end
        end

        %Si el objeto tiene los valores de sus descriptores dentro de
        %unos límites y además no está en el borde de la imagen,
        %entonces se considera como un carácter
        if
            ((area<3000)&&(area>500)&&(extent>0.4)&&(solvidez>0.5)&&(ED<1000)&&(FA<4000)&&(MAL<150)&&(minAL>
            0)&&(perimeter<600)&&(borde==0))
                for i=1:x

```



```
    for j=1:y
        if objeto(i,j)==1
            objeto_F(i,j)=objeto(i,j);
        end
    end
end
end

end

end

%La función devuelve la imagen con todos los objetos identificados como
%un carácter
objetos_detectados=objeto_F;

end
```

```

function reconocimiento_caracteres( imagen )

% La función recibe una imagen binarizada con los caracteres detectados
% previamente en blanco y el fondo en negro. Realiza el reconocimiento de
% cada carácter y los escribe en un fichero .txt

i1=imagen;           %Se carga la imagen
[x,y]=size(i1);     %Se determina el tamaño n° filas y n°columnas
i3=bwareaopen(i1,100); %Se eliminan pequeños objetos con tamaño <100 píxeles
[region numr]=bwlabel(i3,8); %Se realiza el etiquetado y cuenta de objetos

%Carga la matriz de entrenamiento y determina el número de filas
e = load('mc.mat', 'mc');
filas=size(e.mc,1);

fileID=fopen('resultado_OCR.txt','w'); %Se abre fichero .txt como fichero de escritura
for etiqueta=1:numr
    objeto=zeros(x,y); % Se pone en negro la imagen que va a almacenar cada objeto etiquetado

    %La matriz objeto almacena el objeto etiquetado con el valor de etiqueta
    for i=1:x
        for j=1:y
            if region(i,j)==etiqueta
                objeto(i,j)=i1(i,j);
            else
                objeto(i,j)=objeto(i,j);
            end
        end
    end

    %Se calcula la diferencia con cada imagen de entrenamiento
    phi = momentosHu(objeto); %Se calculan los momentos Hu

    %En la matriz resultados se copian las columnas 1,3,4 y 5 de la
    %matriz de entrenamiento. En la columna 2 se almacena la distancia
    %euclídea entre momentos Hu del objeto y momentos Hu de cada fila
    %de la matriz de entrenamiento
    for j=1:filas
        resultados{j,1} = e.mc{j,1};
        resultados{j,2} = norm(phi - e.mc{j,2});
        resultados{j,3} = e.mc{j,3};
        resultados{j,4} = e.mc{j,4};
        resultados{j,5} = e.mc{j,5};
    end

    res_orden = sortrows(resultados,2); %Se ordena matriz de resultados en orden
                                        %ascendente de los valores de la 2ª fila
    props=regionprops(objeto,'all'); %Se extraen las propiedades del objeto
    n_euler=props.EulerNumber; %Se almacena el número de euler

    %Se determina el número de endpoints y branchpoints
    obj_thin=bwmorph(objeto,'thin',Inf);
    EP=bwmorph(obj_thin,'endpoints');
    stats=regionprops(EP,'Area');
    nEP=length(find([stats.Area]>=1));
    BP=bwmorph(obj_thin,'branchpoints');
    stats=regionprops(BP,'Area');
    nBP=length(find([stats.Area]>=1));
    d=props.MajorAxisLength;
    se=strel('disk',round(d/12));
    EP_dilated=imdilate(EP,se);
    stats=regionprops(EP_dilated,'Area');
    nEndPoints=length(find([stats.Area]>10));
    se=strel('disk',round(d/20));
    BP_dilated=imdilate(BP,se);

```

```

stats=regionprops(BP_dilated,'Area');
nBranchPoints=length(find([stats.Area]>10));
n_ep=nEndPoints;
n_bp=nBranchPoints-(nEP-nEndPoints);

%Se recorre la matriz de resultados ordenados hasta encontrar una
%fila en que coincidan n° de euler, n° de endpoints y n° de
%branchpoints con los del objeto, se determina a qué carácter
%corresponde y se escribe en el fichero .txt
i=0;
found=0;
while found<1
    i=i+1;
    if ((res_orden{i,3}==n_euler) && (res_orden{i,4}==n_ep) && (res_orden{i,5}==n_bp))
        found=1;
    end
end
letra=res_orden{i,1};
fprintf(fileID,'%c',letra); %Se escribe el carácter en el fichero .txt de escritura

end
status=fclose(fileID); %Se cierra el fichero .txt con los caracteres reconocidos escritos en él

end

```

```

function reconocimiento_caracteres_thin( imagen )

% La función recibe una imagen binarizada con los caracteres detectados
% previamente en blanco y el fondo en negro. Realiza el reconocimiento de
% cada carácter esqueletizado y los escribe en un fichero .txt

i1=imagen;           %Se carga la imagen
[x,y]=size(i1);     %Se determina el tamaño n° filas y n°columnas
i3=bwareaopen(i1,100); %Se eliminan pequeños objetos con tamaño <100 píxeles
[region numr]=bwlabel(i3,8); %Se realiza el etiquetado y cuenta de objetos

%Carga la matriz de entrenamiento y determina el número de filas
e = load('mc_thin.mat', 'mc');
filas=size(e.mc,1);

fileID=fopen('resultado_OCR.txt','w'); %Se abre fichero .txt como fichero de escritura
for etiqueta=1:numr
    objeto=zeros(x,y); % Se pone en negro la imagen que va a almacenar cada objeto etiquetado

    %La matriz objeto almacena el objeto etiquetado con el valor de etiqueta
    for i=1:x
        for j=1:y
            if region(i,j)==etiqueta
                objeto(i,j)=i1(i,j);
            else
                objeto(i,j)=objeto(i,j);
            end
        end
    end

    %Se calcula la diferencia con cada imagen de entrenamiento
    obj_thin=bwmorph(objeto,'thin',Inf);%Se esqueletiza el objeto
    phi = momentosHu(obj_thin); %Se calculan los momentos Hu

    %En la matriz resultados se copian las columnas 1,3,4 y 5 de la
    %matriz de entrenamiento. En la columna 2 se almacena la distancia
    %euclídea entre momentos Hu del objeto esqueletizado y momentos Hu
    %de cada fila de la matriz de entrenamiento
    for j=1:filas
        resultados{j,1} = e.mc{j,1};
        resultados{j,2} = norm(phi - e.mc{j,2});
        resultados{j,3} = e.mc{j,3};
        resultados{j,4} = e.mc{j,4};
        resultados{j,5} = e.mc{j,5};
    end

    res_orden = sortrows(resultados,2); %Se ordena matriz de resultados en orden
                                        %ascendente de los valores de la 2ª fila

    props=regionprops(objeto,'all'); %Se extraen las propiedades del objeto
    n_euler=props.EulerNumber; %Se almacena el número de euler

    %Se determina el número de endpoints y branchpoints
    obj_thin=bwmorph(objeto,'thin',Inf);
    EP=bwmorph(obj_thin,'endpoints');
    stats=regionprops(EP,'Area');
    nEP=length(find([stats.Area]>=1));
    BP=bwmorph(obj_thin,'branchpoints');
    stats=regionprops(BP,'Area');
    nBP=length(find([stats.Area]>=1));
    d=props.MajorAxisLength;
    se=strel('disk',round(d/12));
    EP_dilated=imdilate(EP,se);
    stats=regionprops(EP_dilated,'Area');
    nEndPoints=length(find([stats.Area]>10));

```

```

se=strel('disk',round(d/20));
BP_dilated=imdilate(BP,se);
stats=regionprops(BP_dilated,'Area');
nBranchPoints=length(find([stats.Area]>10));
n_ep=nEndPoints;
n_bp=nBranchPoints-(nEP-nEndPoints);

%Se recorre la matriz de resultados ordenados hasta encontrar una
%fila en que coincidan n° de euler, n° de endpoints y n° de
%branchpoints con los del objeto esqueletizado, se determina a qué
%carácter corresponde y se escribe en el fichero .txt
i=0;
found=0;
while found<1
    i=i+1;
    if ((res_orden{i,3}==n_euler) && (res_orden{i,4}==n_ep) && (res_orden{i,5}==n_bp))
        found=1;
    end
end
letra=res_orden{i,1};
fprintf(fileID,'%c',letra); %Se escribe el carácter en el fichero .txt de escritura

end
status=fclose(fileID); %Se cierra el fichero .txt con los caracteres reconocidos escritos en él
end

```

```

function [ B ] = bin_umbrales_2( A,x)
%
% Esta función devuelve una matriz B de igual tamaño que A
% con los píxeles con nivel de gris entre 10*x y 10*x+20
% con valor 255 y el resto con valor 0

[f,c]=size(A);
min=10*x;
max=min+20;
if max>256
    max=256;
end
for i=1:f
    for j=1:c
        if ((A(i,j)>=min)&&(A(i,j)<max))
            B(i,j)=255;
        else
            B(i,j)=0;
        end
    end
end
end

end

```

```

function phi = momentosHu(F)

% Esta función calcula los momentos invariantes del objeto presente en la
% imagen de entrada

if (ndims(F) ~= 2) | issparse(F) | ~isreal(F) | ~(isnumeric(F) | ...
    islogical(F))
    error(['F must be a 2-D, real, nonsparse, numeric or logical ' ...
        'matrix.']);
end

F = double(F);
phi = calcular_phi(calcular_eta(calcular_m(F)));

end

```

```

function phi = calcular_phi(e)

%Calcula los momentos phi

phi(1) = e.eta20 + e.eta02;
phi(2) = (e.eta20 - e.eta02)^2 + 4*e.eta11^2;
phi(3) = (e.eta30 - 3*e.eta12)^2 + (3*e.eta21 - e.eta03)^2;
phi(4) = (e.eta30 + e.eta12)^2 + (e.eta21 + e.eta03)^2;
phi(5) = (e.eta30 - 3*e.eta12) * (e.eta30 + e.eta12) * ...
    ( (e.eta30 + e.eta12)^2 - 3*(e.eta21 + e.eta03)^2 ) + ...
    (3*e.eta21 - e.eta03) * (e.eta21 + e.eta03) * ...
    (3*(e.eta30 + e.eta12)^2 - (e.eta21 + e.eta03)^2 );
phi(6) = (e.eta20 - e.eta02) * ( (e.eta30 + e.eta12)^2 - ...
    (e.eta21 + e.eta03)^2 ) + ...
    4 * e.eta11 * (e.eta30 + e.eta12) * (e.eta21 + e.eta03);
phi(7) = (3*e.eta21 - e.eta03) * (e.eta30 + e.eta12) * ...
    ( (e.eta30 + e.eta12)^2 - 3*(e.eta21 + e.eta03)^2 ) + ...
    (3*e.eta12 - e.eta30) * (e.eta21 + e.eta03) * ...
    (3*(e.eta30 + e.eta12)^2 - (e.eta21 + e.eta03)^2 );

end

```

```

function e = calcular_eta(m)

%Calcula los momentos eta

xbar = m.m10 / m.m00;
ybar = m.m01 / m.m00;

e.eta11 = (m.m11 - ybar*m.m10) / m.m00^2;
e.eta20 = (m.m20 - xbar*m.m10) / m.m00^2;
e.eta02 = (m.m02 - ybar*m.m01) / m.m00^2;
e.eta30 = (m.m30 - 3 * xbar * m.m20 + 2 * xbar^2 * m.m10) / m.m00^2.5;
e.eta03 = (m.m03 - 3 * ybar * m.m02 + 2 * ybar^2 * m.m01) / m.m00^2.5;
e.eta21 = (m.m21 - 2 * xbar * m.m11 - ybar * m.m20 + ...
    2 * xbar^2 * m.m01) / m.m00^2.5;
e.eta12 = (m.m12 - 2 * ybar * m.m11 - xbar * m.m02 + ...
    2 * ybar^2 * m.m10) / m.m00^2.5;

end

```

```
function m = calcular_m(F)
```

```
%Calcula los momentos m
```

```
[M, N] = size(F);  
[x, y] = meshgrid(1:N, 1:M);
```

```
x = x(:);  
y = y(:);  
F = F(:);
```

```
m.m00 = sum(F);
```

```
if (m.m00 == 0)  
    m.m00 = eps;  
end
```

```
m.m10 = sum(x .* F);  
m.m01 = sum(y .* F);  
m.m11 = sum(x .* y .* F);  
m.m20 = sum(x.^2 .* F);  
m.m02 = sum(y.^2 .* F);  
m.m30 = sum(x.^3 .* F);  
m.m03 = sum(y.^3 .* F);  
m.m12 = sum(x .* y.^2 .* F);  
m.m21 = sum(x.^2 .* y .* F);
```

```
end
```

```
%Aplicación del método de Otsu
```

```
i1=imread('PRUEBAS.bmp');  
figure(1),imshow(i1);  
i2=rgb2gray(i1); %conversion a escala de grises  
level = graythresh(i2); %determinacion del nivel por el método de Otsu  
BW = im2bw(i2,level); %binarizacion de la imagen según el umbral determinado  
figure(2),imshow(BW);
```