4. IMÁGENES BINARIAS

Las imágenes digitales generalmente están compuestas por un amplio rango de valores de intensidad. A estas imágenes se las ha denominado imágenes de nivel de gris. Aunque el rango de niveles de gris utilizado para representar la imagen tradicionalmente viene siendo de 256 valores (8 bits por píxel), este es variable y depende de la aplicación. La tendencia natural, debido al aumento de la potencia computacional y de la calidad de los sensores es a aumentar este rango para dotar de mayor fidelidad a la imagen, y no parece que esté muy lejos el día en que los 16 bits por píxel (65536 valores) se conviertan en la opción estándar. No obstante, la mayor parte de las aplicaciones no precisan de tantos niveles de gris sino más bien lo contrario; pueden utilizar pocos niveles debido a que trabajan con escenas de muy alto contraste. Tanto es así que en muchas aplicaciones industriales se llega al extremo de utilizar únicamente dos niveles de gris. De esta forma, se obtiene lo que se conoce como imagen binaria.

Trabajar con imágenes binarias resulta muy interesante por dos motivos:

- En primer lugar porque se reduce al mínimo los datos necesarios para representar la imagen y ello permite un máximo aprovechamiento de la potencia computacional.
- En segundo lugar porque las propiedades geométricas y topológicas de los objetos presentes en la imagen, en las que se basan un gran número de aplicaciones industriales, puede obtenerse rápida y fácilmente a partir de las imágenes binarias.

Desde el punto de vista computacional, las imágenes binarias se procesan mucho más rápidamente. Por este motivo, las primeras aplicaciones de visión artificial en línea emplearon este tipo de imágenes casi exclusivamente. Incluso hoy en día, que se dispone de potentes procesadores para hacer frente a imágenes en niveles de gris, siguen siendo aún más numerosas las aplicaciones que trabajan sobre imágenes binarias por su simplicidad y robustez. Dado el protagonismo que tienen en el ámbito industrial las dedicaremos un capítulo en exclusiva.

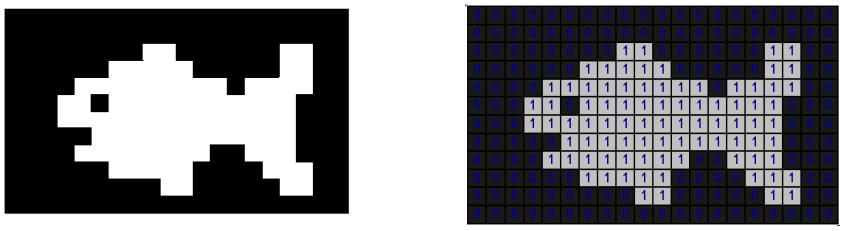


Figura 4.1. Las imágenes binarias solo están representadas por dos niveles de gris: el negro y el blanco. Cada píxel está etiquetado con 0 o 1 respectivamente.

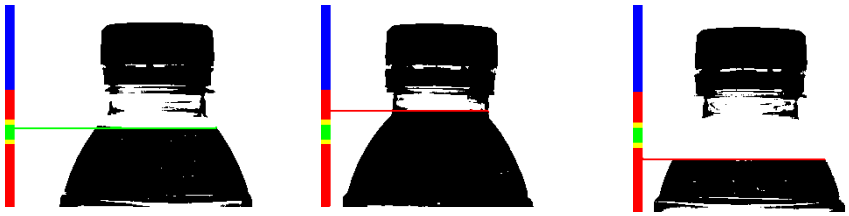


Figura 4.2. La inspección del nivel de llenado en botellas es una de las muchas aplicaciones industriales que hacen uso de imágenes binarias. La binarización permite con mínimos recursos computacionales inspeccionar más de 50.000 botellas a la hora ya que reduce al máximo la representación de la imagen.

4.1 BINARIZACIÓN

Las imágenes binarias siempre se obtienen a partir de imágenes de niveles de gris. En la actualidad no existen cámaras comerciales que proporcionen imágenes binarias. El proceso de conversión de una imagen de nivel de gris a una imagen formada solo por dos valores o etiquetas (0 para el negro y 1 para el blanco) se conoce como *binarización*.

Para introducir el procedimiento de binarización consideremos la imagen que aparece en la figura 4.3, que ha sido obtenida iluminando la pieza a contraluz. El histograma de esta imagen presenta dos poblaciones claramente diferenciadas: una de píxeles con muy altos niveles de gris pertenecientes al fondo retroiluminado y otra de píxeles con bajos valores correspondientes al objeto, ya que este no recibe luz en la cara que muestra a la cámara.

Este marcado carácter bimodal del histograma nos permitirá clasificar los píxeles de la imagen en fondo y objeto con total garantía sin más que atender a su nivel de gris. Solo será preciso establecer un umbral de división en el nivel de gris: todos los píxeles con un nivel de gris por debajo de ese umbral, que podemos fijar por ejemplo en mitad de la escala *(128)*, corresponden al objeto y los que superan este umbral son del fondo. El resultado se muestra en la figura 4.3c, donde solamente existen píxeles negros y blancos, que representarían respectivamente al objeto y al fondo de la imagen. Esta técnica recibe el nombre de *binarización* por originar una imagen binaria.

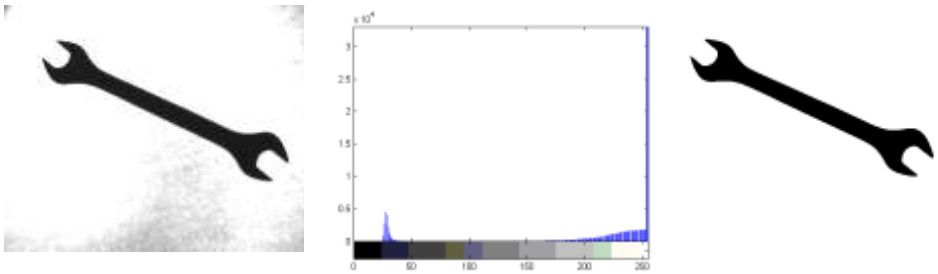


Figura 4.3. La binarización funciona muy bien en escenas sencillas muy contrastadas como las obtenidas a contraluz. De hecho esta técnica ha sido una de las más utilizadas a nivel industrial.

La binarización tiene una gran utilidad en procesamiento automático de imagen pues reduce enormemente la cantidad de datos de la imagen de una forma muy sencilla. Si se parte de imágenes bien contrastadas, la binarización permite con muy poco procesamiento un análisis fiable de la imagen. Para la obtención de imágenes de gran contraste se suele recurrir a la utilización de técnicas de retroiluminación (contraluz). Estas imágenes obtenidas a contraluz pueden transformarse sin pérdida significativa de información a binarias.

La forma más inmediata de representar una imagen de niveles de gris mediante una imagen binaria es considerar únicamente el bit más significativo del nivel de gris de cada píxel. O lo que es lo mismo, fijar un umbral en la mitad de la escala de gris que servirá de referencia para asignar en la imagen binaria un 0 si el nivel de gris del píxel es inferior o un 1 si supera este umbral. Por ejemplo, en una imagen típica de 256 niveles representada por 8 bits/píxel tomar el primer bit es lo mismo que poner a 0 (negro) todo aquel píxel que presente un nivel de gris por debajo de 128 y a 1 (blanco) el resto.

Aunque el dividir la escala de grises en dos partes iguales para asignar 0 o 1 puede parecer la forma más correcta de binarizar, en realidad la mayoría de las veces esto no es así. El rango dinámico de una imagen no siempre se extiende a todo el intervalo de niveles de gris posible y, aun así, muchas veces es más adecuado fijar el umbral de binarización en otro punto distinto del valor medio de la escala de grises.

En definitiva, la binarización consiste en, a partir de un nivel de gris predeterminado denominado *umbral de binarización*, etiquetar como 0 todos los píxeles con niveles de gris inferior a ese umbral y como 1 los que tengan un nivel de gris igual o superior.

Si la imagen está bien contrastada, la pérdida de información es mínima. Muchas veces esta sencilla operación permite separar los objetos del fondo. Hay que insistir en que, para ello, es fundamental que la imagen de nivel de gris tenga un alto contraste, es decir, que los dos grupos de píxeles correspondiente a objetos y al fondo, posean niveles de gris bien diferenciados.

Figura 4.4. Muchas aplicaciones de guiado de robots operan sobre imágenes binarias. Las imágenes binarias contienen toda la información referente a la silueta del objeto y por tanto de ellas se puede extraer rápidamente la localización y orientación del objeto a capturar.

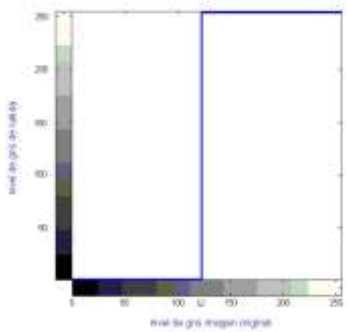


Figura 4.5. Binarización: representación gráfica de la transformación. A los niveles por debajo del umbral de binarización *U* se les asigna el negro y a los demás el blanco.

4.1.1 Elección del umbral de binarización

Precisamente, la etapa clave en la binarización es la elección del valor *umbral*, ya que este constituirá la referencia para separar el fondo claro del objeto oscuro, o viceversa. La separación debería poder hacerse de forma ideal si se conociese la distribución de los píxeles oscuros y claros. En general, estas distribuciones no son conocidas de antemano, pero con el histograma de la imagen se obtiene la información del número de píxeles relativa a cada nivel de gris, información que resulta muy valiosa.

Si las distribuciones de los niveles de gris asociados a los píxeles claros y oscuros están muy separadas, entonces el histograma será bimodal, como el que se puede observar en la figura 4.3. El umbral más idóneo será el que divida los niveles de gris en dos poblaciones bien diferenciadas. El umbral óptimo estará en el valle del histograma.

```
→ I:        Imagen original de niveles de gris de tamaño (filas,columnas)
  umbral:   Umbral de binarización
← B:        Imagen binaria
BINARIZACIONFIJA(I,umbral)
    Inicializar B=0                    %Se inicializan a 0 todos los valores de la matriz
    Mientras se recorre I(x,y)  x=1…filas,  y=1…columnas
        Si I(x,y) > umbral                          %Presuponemos fondo obscuro
            B(x,y) = 1
        Fin
    Fin
    Devolver B
```

Algoritmo 4.1: Binarización con umbral fijo

Por el contrario, cuando las distribuciones comienzan a estar más solapadas, la elección del umbral como un valor perteneciente al valle empieza a ser difícil, puesto que el valle ya no aparece tan claro. En este tipo de imágenes la binarización no proporcionará buenos resultados.

A la hora de establecer el umbral de binarización en una aplicación industrial pueden utilizarse dos estrategias: preestablecer un umbral fijo para todas las imágenes capturadas o bien calcular en cada imagen el umbral idóneo.

4.1.2 Binarización con umbral fijo

La técnica más utilizada en aplicaciones industriales, debido a que el coste computacional para determinarlo es nulo, es elegir un umbral fijo para binarizar todas las imágenes que se capturan en el proceso. Este valor fijo puede obtenerse analizando previamente los histogramas de las imágenes obtenidas durante la fase de desarrollo de la aplicación. Esta estrategia debe ser implantada únicamente en entornos controlados, con una iluminación muy estable. Variaciones en la iluminación de la escena pueden ocasionar cambios en los niveles de gris de la imagen que invaliden la idoneidad del umbral fijado.

Dado que la zona para establecer la división será el valle del histograma, cuanto más ancho sea este valle, más fiable va a ser trabajar con un umbral fijo, puesto que existirá un mayor margen de error. La anchura del valle puede absorber las oscilaciones en los niveles de gris que pueda haber por variaciones en la iluminación y hacer que el sistema funcione perfectamente aun cuando existan estas perturbaciones. Una buena configuración del sistema de iluminación permitirá aumentar al máximo la distancia entre los dos picos que aparecen en el histograma correspondiente al objeto y al fondo.

El algoritmo de binarización con umbral fijo, Algoritmo 4.1, presupone que los objetos de interés aparecerán con niveles de gris más claros que el fondo. En caso contrario, una sencilla operación de inversión podría llevarse a cabo como paso siguiente a la binarización.

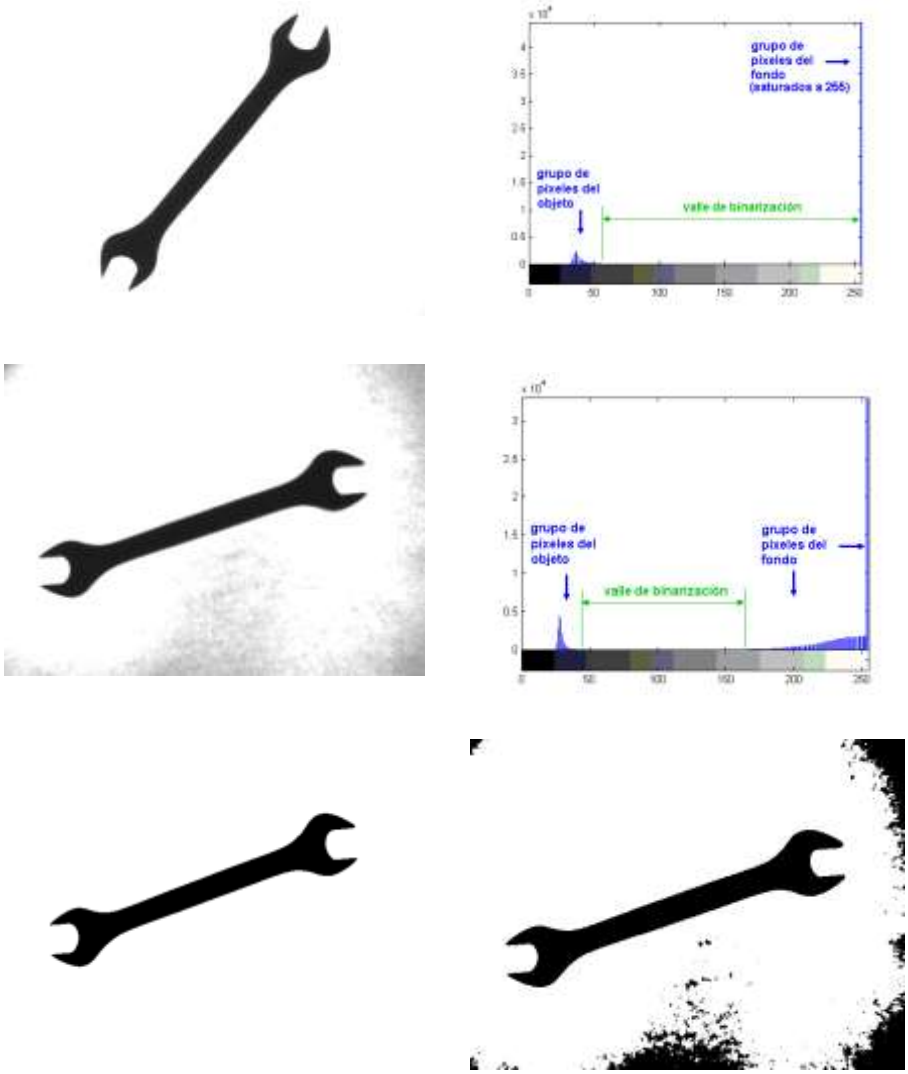


Figura 4.6. a) Con una buena retroiluminación se logran imágenes muy contrastadas, con histogramas que presentan una gran separación entre las dos poblaciones de gris. b) La anchura del valle de su histograma es de 200 niveles de gris. c) Imagen obtenida en el mismo sistema pero con una degradación de la iluminación. d) Apréciese cómo se modifica el histograma estrechándose el valle. e) Resultado de binarizar la imagen en c) con un umbral de 150. La imagen binaria todavía es buena pero el umbral prefijado se encuentra en el mismo límite del valle. f) Una pérdida adicional de luz haría que la binarización proporcionara una imagen de mala calidad como la presentada en la figura. Con una binarización automática, el umbral se habría reajustado a un valor más bajo, ubicándole dentro del valle del histograma, y el resultado hubiera sido una imagen binaria limpia como la de la imagen e).

```
→ I:  Imagen original de G niveles de gris de tamaño (filas,columnas)
← B:  Imagen binaria
BINARIZACIONAUTOMATICA(I)
    hist=HISTOGRAMA(I)
    Mientras se recorre hist(k),  k=1… G
        q1=∑_{i=1}^{k} hist(i)                          %número de píxeles en 1er grupo
        q2=∑_{i=k+1}^{N} hist(i)                        %número de píxeles en 2º grupo
        var1= VARIANZA({hist(1),…, hist(k)})            %varianza 1er grupo
        var2= VARIANZA({hist(k+1),…, hist(G)})          %varianza 2º grupo
        suma= q1·var1 + q2·var2                         %suma ponderada varianzas
        Si k==1                                         %inicializa variables en 1ª iteración
            menor_suma=suma
            umbral_optimo=1
        en caso contrario
            Si suma < menor_suma                        %actualiza umbral si óptimo
                menor_suma=suma
                umbral_optimo=k
            Fin
        Fin
    Fin
    B=BINARIZACIONFIJA(I,umbral_optimo)
    Devolver B
```

Algoritmo 4.2: Binarización con selección automática del umbral

4.1.3 Binarización automática

Cuando se prevé que la iluminación va a estar expuesta a variaciones, no se puede trabajar con un umbral fijo. En este caso, lo apropiado es hallar para cada imagen, utilizando su histograma, el valor más idóneo como umbral.

El objetivo de esta técnica será calcular automáticamente, para cada imagen de niveles de gris que aparezca en el proceso, el umbral de binarización óptimo. Así, se pueden evitar los problemas de utilizar un umbral fijo, tales como el desplazamiento del histograma debido a cambios en la iluminación.

Un algoritmo sencillo de elección de un umbral automático para un histograma bimodal es el propuesto por (Otsu, 1979). La idea es calcular iterativamente el valor de nivel de gris medio y su varianza para cada uno de los dos modos del histograma, situados uno entre 0 y k y el otro entre k y G, calculando además la suma ponderada de las varianzas. El algoritmo escoge como umbral óptimo el valor para el que esa suma ponderada es mínima.

El pseudocódigo para la binarización automática se muestra en el algoritmo 4.2. Este algoritmo puede optimizarse de forma sencilla (Haralick, 1993), pero la implementación presentada muestra más claramente la esencia del algoritmo.

No hay que olvidar que, para obtener resultados aceptables en una binarización global, las imágenes deben presentar un histograma bimodal, es decir, con dos poblaciones de píxeles con niveles de gris claramente diferenciados. Sobre imágenes con histogramas relativamente planos no se puede obtener una buena representación con dos niveles de gris aunque se emplee una búsqueda automática del umbral.

En imágenes procedentes de aplicaciones con histogramas más complejos, pueden emplearse técnicas más elaboradas de binarización automática, como aquellas en las que el umbral óptimo varía localmente sobre la imagen u otras que utilizan varios umbrales de binarización. No las abordaremos en este documento por ser menos habituales en escenas industriales. No obstante, para el lector interesado, una revisión de técnicas de binarización automática puede encontrarse en (Sezgin, 2009).

4.2. DEFINICIONES TOPOLÓGICAS BÁSICAS

La extracción de los objetos de interés de una imagen binaria, técnica conocida como segmentación, asume que los puntos pertenecientes a los objetos de interés están *cercanos* entre sí. Por ello, es necesario introducir algunas definiciones básicas que formalicen este concepto de proximidad espacial.

Véase por ejemplo (Kong, 1989) para una presentación más detallada.

4.2.1 Vecindad

Un píxel P=*(x,y)* de la imagen tiene 2 vecinos horizontales (este y oeste) y 2 verticales (norte y sur). Las coordenadas de estos píxeles son *(x+1,y), (x-1,y), (x,y+1)* y *(x,y-1)*. Este conjunto de píxeles son los 4-vecinos $\mathcal{N}_4$(P) de *(x,y)* que puede verse en la figura 4.7.a.

Los 4 vecinos situados en las diagonales $\mathcal{N}_D$(P) están en las coordenadas *(x-1,y-1), (x-1,y+1), (x+1,y-1)* y *(x+1,y+1)*. El conjunto $\mathcal{N}_D$(P) U $\mathcal{N}_D$(P) formado por la unión de los anteriores constituye la 8-vecindad $\mathcal{N}_8$(P) del píxel *(x,y)* que se ha representado en la figura 4.7.b.

4.2.2 Adyacencia

Dos píxeles son adyacentes si:

- Son vecinos (por ejemplo $\mathcal{N}_4$(P), $\mathcal{N}_8$(P), …)
- Sus valores satisfacen algún criterio específico de similitud, típicamente la igualdad.

Notaremos como $\mathcal{V}$ el conjunto de valores de los píxeles usados para definir la similitud: por ejemplo, los píxeles que tras una binarización tienen el valor 1, tienen a 1 como valor de similitud**,** $\mathcal{V}$={1}.

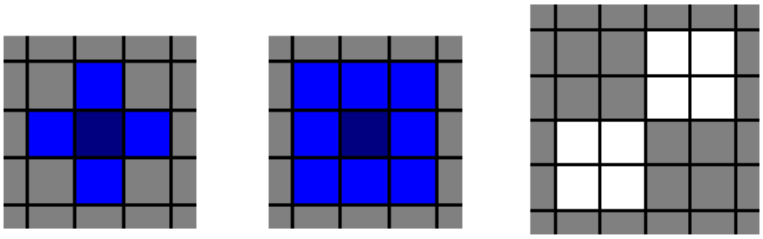


Figura 4.7. a) Píxeles considerados vecinos de P(x,y) en la 4-vecindad

$$\mathcal{N}_4(P) = \{(x,y-1)\ ,\ (x,y+1),\ (x-1,y),\ (x+1,y)\}$$

b) En la 8-vecindad se incluyen además como vecinos los píxeles en las diagonales

$$\mathcal{N}_8(P) = \mathcal{N}_4(P) \cup \{(x-1,y-1)\ ,\ (x-1,y+1),\ (x+1,y-1),\ (x+1,y+1)\ \}$$

c) En esta porción de imagen se considerarán una o dos componentes conexas dependiendo del tipo de adyacencia que se haya predefinido. Si se considera 8-adyacencia, los píxeles en contacto únicamente por el vértice son vecinos y formarían parte de una misma región de 8 píxeles. Al contrario, si se considera 4-adyacencia pertenecerían a dos regiones distintas, de 4 píxeles cada una.

Consideraremos 2 tipos básicos de adyacencia:

- *4-adyacencia*: 2 píxeles P=*(x,y)* y Q=*(w,z)* con *I(x,y)* y *I(w,z)* $\in \mathcal{V}$, son 4-adyacentes si P $\in \mathcal{N}_4$(Q)
- *8-adyacencia*: La definición es análoga a la de la 4-adyacencia sustituyendo la 4-vecindad $\mathcal{N}_4$(P) por la 8-vecindad $\mathcal{N}_8$(P)

4.2.3 Camino

Un camino de longitud *n* desde el píxel $P_0 = (x_0,y_0)$ al píxel $P_n = (x_n,y_n)$ es una secuencia de píxeles distintos P_0, P_1, …, P_n donde P_i es adyacente a P_{i-1} para $1 \leq i \leq n$.

Si $P_0 == P_n$ el camino es cerrado. Dependiendo del tipo de adyacencia utilizado, tendremos 4-caminos, 8-caminos, etc.

Dos píxeles P y Q están *conectados* si existe un camino entre ellos.

4.2.4 Regiones y contornos

Para un conjunto *S* de píxeles:

- Dado un píxel P $\in S$, el subconjunto de todos los píxeles $Q_i \in S$ conectados a P forman una *componente conexa* de *S*.
- Si *S* solo tiene una componente conexa, *S* es un *conjunto conexo*.

Un subconjunto $\mathcal{R}$ de píxeles de una imagen es una *región* si $\mathcal{R}$ es un conjunto conexo.

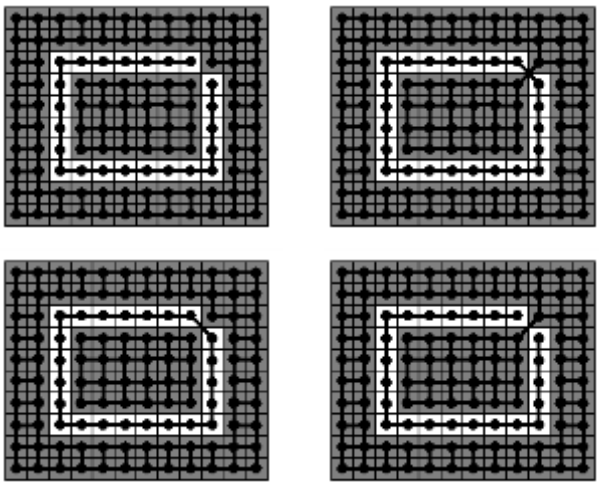


Figura 4.8. En los casos a) y b) vemos que se entra en contradicción con el Teorema de la curva de Jordan. a) Si consideramos 4-adyacencia para todos los píxeles de la imagen observamos que la *curva digital abierta* en blanco ha generado dos regiones inconexas. b) Si consideramos 8-adyacencia para todos los píxeles de la imagen observamos que la *curva digital cerrada* no ha generado dos regiones inconexas. c) y d) En estos casos, cuando elegimos distintos tipos de adyacencia para la curva digital en blanco respecto del fondo obscuro, tenemos 2 regiones si esta está cerrada c) o 1 si está abierta d).

El *contorno* o *frontera* de una región $\mathcal{R}$ es la curva digital formada por el conjunto de píxeles P $\in \mathcal{R}$ que tienen al menos un vecino Q tal que Q $\notin \mathcal{R}$.

Un problema importante que debe comentarse al trabajar con curvas es el relativo al Teorema de la curva de Jordan y su extensión a imágenes digitales. Según este teorema, una curva cerrada C divide los puntos de la imagen no pertenecientes a C en dos conjuntos distintos sin puntos en común (*conjunto interior y conjunto exterior*) para los cuales esta curva es su *frontera común*. Es fácil comprobar que si establecemos el mismo tipo de adyacencia para todos los píxeles de la imagen, este teorema no se cumple, lo que tiene serias consecuencias a la hora de desarrollar los algoritmos. Para evitar este problema, basta con establecer un tipo de adyacencia diferente para los píxeles correspondientes al fondo (0) respecto a los objetos (1), de tal forma que si elegimos 4-adyacencia para el fondo, elegiremos 8-adyacencia para los objetos, y viceversa (Rosenfeld, 1975).

4.3. ETIQUETADO DE COMPONENTES CONEXAS

Entre las regiones presentes en una imagen binaria, es lógico suponer que se encuentren aquellas que corresponden con los objetos de interés en la escena, sin más que atender a un criterio de proximidad espacial.

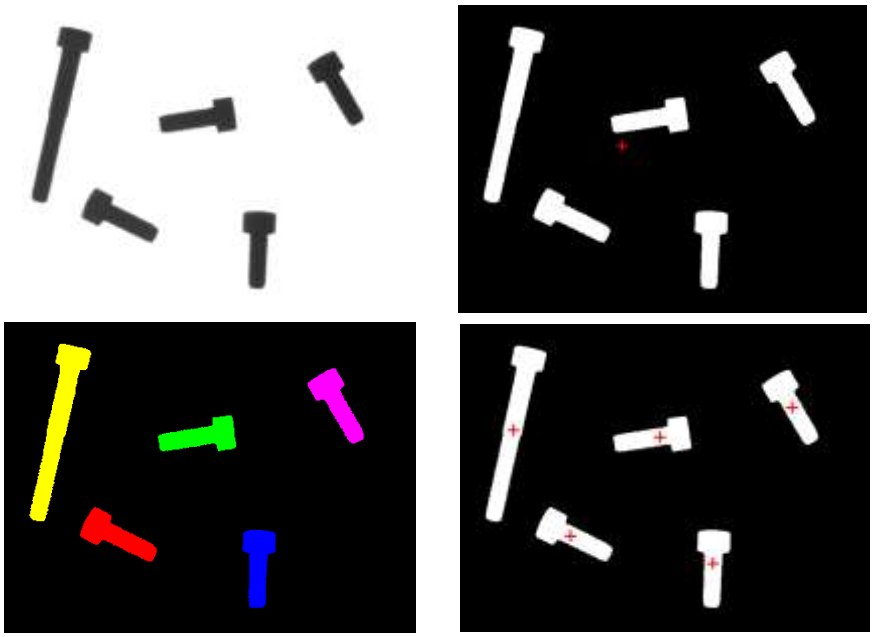


Figura 4.9. Si se trata de extraer el área o el centro de gravedad a partir de la simple binarización de la imagen original se computarán todos los píxeles blancos como si constituyesen parte de un mismo objeto. En la imagen el área total es de 24581 píxeles y el centro de gravedad del conjunto se hallaría en la posición representada por la cruz de la figura. Como las aplicaciones generalmente precisan de una extracción individualizada de las características de cada uno de los componentes presentes en la imagen, es preciso previamente establecer de alguna forma cada una de las regiones o componentes conexas que en ella aparecen. Esta operación es lo que se conoce como etiquetado. En la imagen c) cada etiqueta se ha representado por un color. El etiquetado permite extraer las características individuales de cada objeto estableciendo las áreas individuales en píxeles. Estas áreas individuales ya permiten establecer ciertos criterios para detectar o reconocer determinados objetos. Por ejemplo, se podría establecer que existe un elemento significativamente más grande que el resto y que este tiene su centro de gravedad en (84.63 , 158.44).

Para poder identificar los píxeles que pertenecen a cada una de las regiones y, por tanto, asociar estas regiones a los objetos para extraer sus características, es necesario obtener las *componentes conexas* presentes en la imagen. Es obvio que, si en la imagen aparecen varias regiones, no se podrá obtener, por ejemplo, su área o centro de gravedad, hasta que estas no estén identificadas de forma individual. Este proceso se conoce como *etiquetado* y consiste en definitiva en asignar una misma etiqueta a cada uno de los píxeles que pertenecen a una componente conexa. La etiqueta será un valor numérico que permite identificar todos los píxeles que forman parte de una misma región. De esta forma, la imagen de etiquetas obtenida permitirá la extracción individualizada de características de cada una de las regiones conexas de la imagen.

La definición de un componente conexo dependerá del tipo de adyacencia utilizado. Si utilizamos la 4-adyacencia, las regiones resultantes serán *4-conexas* y si utilizamos la 8-adyacencia las regiones serán *8-conexas*.

El etiquetado de imágenes ha sido un campo de gran actividad investigadora, siendo muy numerosos y diversos los enfoques para su solución (Grana, 2009). Presentaremos por razones didácticas solo algunos algoritmos básicos de etiquetado.

4.3.1 Algoritmo iterativo de etiquetado

Seguidamente se expone en el Algoritmo 4.3 un sencillo procedimiento iterativo para llevar a cabo el etiquetado de las componentes conexas (Haralick, 1993). Este algoritmo no se caracteriza precisamente por su bajo tiempo de procesamiento, pero permite de una forma sencilla entender la base del problema de etiquetado.

Consiste en una etapa de inicialización seguida de diversas etapas en las que se alterna un barrido hacia abajo de izquierda a derecha seguido de un barrido hacia arriba de derecha a izquierda de la imagen.

En la etapa de inicialización se crean tantas etiquetas como píxeles en blanco tiene la imagen. La forma más sencilla de llevar a cabo esta inicialización es simplemente barrer la imagen e ir numerando los píxeles blancos que vayan apareciendo.

El paso siguiente consiste en recorrer de forma iterativa la imagen previamente etiquetada dos veces, una hacia abajo y de izquierda a derecha y otra hacia arriba de derecha a izquierda. En estos barridos, para cada píxel etiquetado se analizará su vecindad de tal forma que se actualizará su etiqueta si entre los vecinos existe uno con un valor de etiqueta inferior.

El proceso de doble barrido continuará mientras se produzca algún cambio de etiqueta en cualquiera de los dos barridos.

Como se apunta en la función ***ACTUALIZAETIQUETA()***, debe notarse que no es necesario analizar todos los vecinos (Suzuki, 2003). Supongamos que deseamos componentes 4-conexas. En el barrido hacia abajo de izquierda a derecha solo es necesario analizar los píxeles situados al norte y al oeste, pues son los únicos que han podido sufrir una modificación en el actual barrido. De la misma forma, en el barrido hacia arriba de derecha a izquierda solo será necesario analizar el valor de la etiqueta de los píxeles situados al este y al sur

Al finalizar el algoritmo, las regiones conexas aparecerán etiquetadas con valores no consecutivos. Esto se resuelve con un nuevo barrido de la imagen que reasigne el valor de las etiquetas, función ***REASIGNAETIQUETAS()***.

Insistimos en que este algoritmo iterativo no se caracteriza por su eficiencia, pero ilustra perfectamente el proceso de etiquetado.

→ *B*: Imagen binaria (0 = Fondo, 1 = Objetos) de tamaño *(filas,columnas)*
a: Tipo de adyacencia, 4 u 8
← *E*: Imagen de etiquetas

ETIQUETADOITERATIVO(B,a)
Inicializar contador *cont*=0 y matriz de etiquetas, *E*=**0**
Mientras se recorre *B(x,y)* *x=1…filas, y=1…columnas*
Si *B(x,y)* == 1 %Si es píxel blanco
cont=cont+1
E(x,y) = cont %Vamos numerando los píxeles blancos
Fin
Fin %Todos píxeles blancos ya están numerados
cambio=1
Mientras *cambio==1*
cambio=0
b=1 %Simboliza barrido hacia abajo de izqda. a dcha.
Mientras se recorre *E(x,y)* según *b* *x=1…filas, y=1…columnas*
Si *E(x,y)>0* %Solo miramos positivos. *E(x,y)*=0 en fondo
*cambio=**ACTUALIZAETIQUETA**(E,x,y,a,b)*
Fin
Fin
b=2 %Simboliza barrido hacia arriba de dcha. a izqda.
Mientras se recorre *E(x,y)* según *b* *x=fi1as…1, y=columnas…1*
Si *E(x,y)>0* %Solo miramos positivos. *E(x,y)*=0 en fondo
*cambio=**ACTUALIZAETIQUETA**(E,x,y,a,b)*
Fin
Fin
Fin
REASIGNAETIQUETAS(E)
Devolver *E*

→ *E*: Matriz actual de etiquetas. Se modifica dentro de la función
x, y: Coordenadas del píxel actual
a: Tipo de adyacencia, 4 u 8
b: Tipo de barrido (1 hacia abajo, 2 hacia arriba) que decide vecinos a visitar
← *cambio*: 0 si no ha habido cambio, 1 en caso contrario

%Función que sustituye la etiqueta en *(x,y)* si existe un valor menor en la vecindad.

ACTUALIZAETIQUETA(E,x,y,a,b)
cambio=0
Obtener puntos $\mathcal{V}=\{(x_1,y_1)\ldots(x_k,y_k)\}$, $\mathcal{V}\subset\mathcal{N}_{a,b}(x,y)$ *AND* $E(x_i,y_i)>0$, $i=1\ldots k$
*m= **MINIMO**(*$E(x_1,y_1)$, …, $E(x_k,y_k)$*)*
Si *E(x,y)>m*
E(x,y)=m
cambio=1
Fin
Devolver *cambio*

Algoritmo 4.3: Etiquetado iterativo

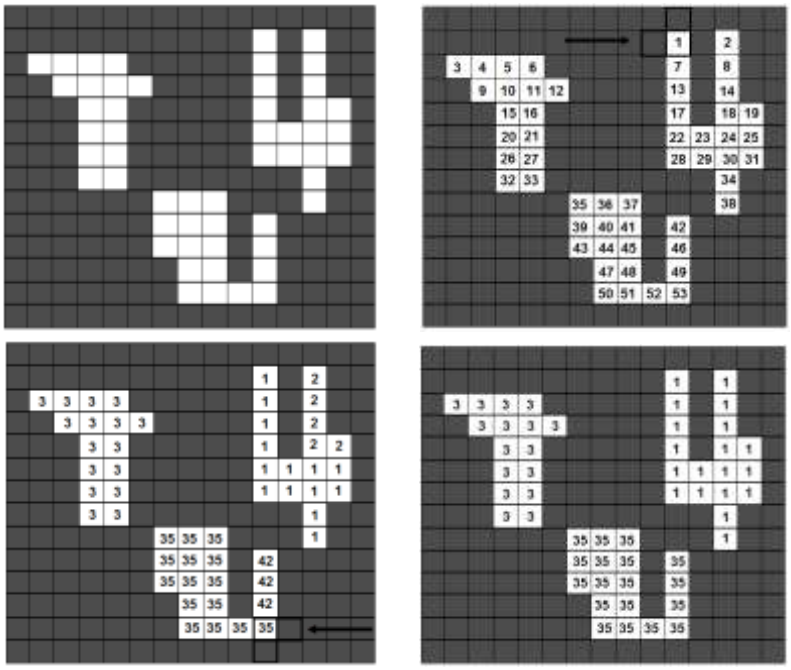


Figura 4.10. Algoritmo de etiquetado iterativo. En primer lugar se inicializa la imagen a) numerando todos los píxeles blancos. A continuación, haciendo un barrido de arriba abajo y de izquierda a derecha, se actualiza cada píxel si su etiqueta es mayor que el valor mínimo de etiqueta que existe en su vecindad. En la figura se ha considerado 4-adyacencia. c) Finalizado el barrido de arriba abajo y de izquierda a derecha se procede análogamente pero de abajo a arriba y de derecha a izquierda. d) Imagen de etiquetas *E* obtenida. Cada objeto que aparecía en la imagen binaria en blanco se muestra ahora con una etiqueta exclusiva. Debe tenerse en cuenta que este algoritmo necesita un proceso que reasigne las etiquetas ya que las obtenidas no son consecutivas. Por ejemplo, en esta imagen con tres objetos, un objeto tiene la etiqueta 1, otro la 3 y otro la 35.

4.3.2 Algoritmo recursivo de etiquetado

Un algoritmo conceptualmente simple de etiquetado consiste en ir *rellenando* cada una de las regiones a partir de un primer píxel que actúa como *semilla*. Existen un amplio abanico de variantes de este tipo de algoritmos (*flood* o *seed fill*) en el ámbito de los gráficos por ordenador (Lieberman, 1978).

Su implantación más simple consiste en barrer la imagen hasta encontrar el primer píxel blanco de un objeto. Este píxel se pondrá a negro en la imagen binaria y se etiquetará a 1 en la imagen de etiquetas. A continuación, y de forma recursiva, buscaremos en la vecindad de cada píxel previamente etiquetado otro píxel blanco, que pondremos a 0 en la imagen binaria y a 1 en la de etiquetas. Al cabo de un tiempo, todos los píxeles de la región habrán sido visitados y tendrán un valor 0 en la imagen binaria, completándose el rellenado. El proceso continuaría buscando un segundo píxel blanco, que se etiquetaría como 2, etc.

→ *B*: Imagen binaria (0 = Fondo, 1 = Objetos) de tamaño *(filas,columnas)*
a: Tipo de adyacencia, 4 u 8
← *E*: Imagen de etiquetas
netiq: Número de componentes conexas
area: Vector con el área de cada componente conexa

***ETIQUETADORECURSIVO** (B,a)*
Inicializar *netiq* =0, imagen de etiquetas *E*=**0**, *area* =**0** y la pila $\mathcal{P}=\phi$
Mientras se recorre *B(x,y)* *x=1…filas, y=1…columnas*
Si *B(x,y)* == 1
netiq=netiq+1
area(netiq)=0
Añadir coordenadas *(x,y)* a la pila $\mathcal{P}$
B(x,y) = 0
Mientras $\mathcal{P}\neq\phi$ % Mientras haya algún píxel *(x,y)* en la pila
Remover coordenadas *(x,y)* de la pila $\mathcal{P}$
E(x,y) = netiq
area(netiq)=area(netiq)+1
Para cada $(x_i,y_i) \in \mathcal{N}_a(x,y)$
Si $B(x_i,y_i)$ == 1
Añadir coordenadas (x_i,y_i) a la pila $\mathcal{P}$
$B(x_i,y_i) = 0$
Fin
Fin
Fin
Fin
Fin
Devolver *E, netiq, area*

Algoritmo 4.4: Etiquetado recursivo

Este algoritmo, a diferencia del anterior, proporciona de forma directa las etiquetas numeradas en forma consecutiva y, además, calcula el número de píxeles de cada región.

En el algoritmo 4.4 se presenta el pseudocódigo de esta técnica usando una estructura de datos tipo *pila* (*último en entrar, primero en salir*) para almacenar las coordenadas de los vecinos a visitar y así evitar el uso de funciones recursivas. De esta forma, para cada región, estamos realizando un recorrido de un *árbol en profundidad*, donde el nodo raíz es el primer píxel semilla, el siguiente nivel del árbol está formado por los nodos hijos que son vecinos al nodo semilla, etc.

Obviamente, el algoritmo puede implantarse también mediante un recorrido de un *árbol en anchura*, sustituyendo la pila por una *cola* (*primero en entrar, primero en salir*).

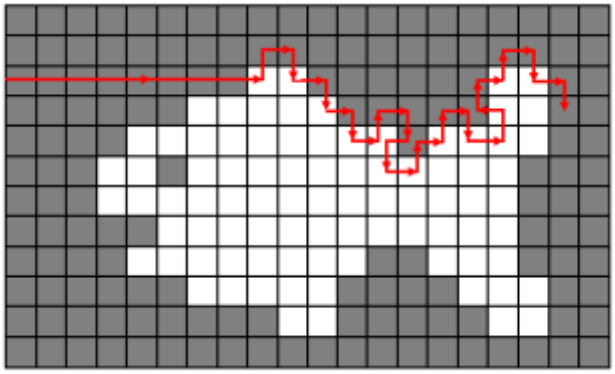


Figura 4.11. Algoritmo *square tracing* o *tortuga de Papert***.** Si encuentra un píxel del objeto gira a su izquierda. Si no encuentra un píxel del objeto gira a su derecha.

4.4. SEGUIMIENTO DE CONTORNOS EN IMÁGENES BINARIAS

Los *contornos* o *fronteras* de una componente conexa contienen información muy valiosa, pues proporcionan la silueta de los objetos. Así, se podrá disponer de datos como la longitud del contorno del objeto que junto con el área nos dará una idea de la forma (compacidad o circularidad).

Un punto P pertenece al contorno C de una región $\mathcal{R}$ si existe un punto Q 4-vecino en el conjunto complementario $\mathcal{R}^c$: P ∈ C si existe Q ∈ ($\mathcal{R}^c \cap \mathcal{N}_4$(P)).

En general, interesa obtener los contornos como una *secuencia ordenada* de puntos 8-conectados, dado que identificar los puntos del contorno de una región de forma aislada es simple pero de poca utilidad, salvo quizás a efectos gráficos. Entre los muchos algoritmos propuestos en la bibliografía **(**Ghuneim, 2012), uno de los más sencillos es el conocido como *square tracing* o *tortuga de Papert* (Papert, 1973) que, aunque presenta fallas en determinadas configuraciones, da una idea de la operativa de este tipo de algoritmos.

La forma de operar de este algoritmo es realizar un barrido de la imagen hasta encontrar un primer píxel de la región, que por fuerza pertenece al contorno. Este punto actúa como semilla del seguimiento *square tracing*. El proceso puede ser descrito en términos de un observador que camina sobre los píxeles de la imagen. Si el observador visita un píxel de la región se desplazará a su izquierda (es preciso por tanto tener en cuenta la dirección de avance del observador). Si por el contrario, el píxel que alcanza el observador corresponde al fondo y no a la región, entonces se desplazará hacia la derecha en el siguiente movimiento. El seguidor seguirá este protocolo hasta llegar al píxel inicial. Sin embargo, para no dar por terminado prematuramente el seguidor, por ejemplo debido a la presencia de zonas de espesor un píxel, debe exigirse que la visita al píxel inicial se haga en la misma dirección en la que se hizo al principio.

```
→ E:              Imagen etiquetada de tamaño (filas,columnas)
  P_inicial=(x,y): Coordenadas del punto inicial
  etiq:           Etiqueta de la región
  dir_inicial:    Direcciones de avance del seguidor (0 hacia la derecha, 1 hacia abajo.
                  2 hacia la izquierda y 3 hacia arriba)
← contorno:       Vector con coordenadas (x_i,y_i) del contorno de la región etiq
SQUARETRACING(E,x,y,etiq,dir_inicial)
    Inicializar contorno=0
    dir=dir_inicial
    num=1                                          %Contador de puntos del contorno
    contorno(1,:)=(x,y)   %Añade coordenadas iniciales P_inicial=(x,y),en la posición 1
    P=(x,y)=GIROIZQUIERDA(P_inicial=(x,y),dir)
    dir=MOD(dir+3,4)                    %MOD() Función resto de la división entera
    Mientras P ≠ P_inicial OR dir≠ dir_inicial
        Si E(x,y)==etiq
            num=num+1
            contorno(num,:)=(x,y)   %Añade coordenadas (x,y) en la posición num
            P=(x,y)=GIROIZQUIERDA(P)
            dir=MOD(dir+3,4)
        en caso contrario
            P=(x,y)=GIRODERECHA(P)
            dir=MOD(dir+1,4)
        Fin
    Fin
    Devolver contorno
```

Algoritmo 4.5: Seguidor de contornos *square tracing* o *tortuga de Papert*

El algoritmo *square tracing*, cuyo pseudocódigo se presenta en el algoritmo 4.5, y algunas variantes mejoradas, tienen el inconveniente de que no funcionan de manera directa en imágenes binarias en las cuales las regiones tienen agujeros: así como una región solo tiene una frontera externa, tendrá tantas fronteras internas como agujeros.

4.4.1 Seguidor de contornos y etiquetado de regiones simultáneos

El algoritmo propuesto en (Chang 2004) tiene la característica de no solo obtener el contorno exterior y los contornos interiores de una región, sino que simultáneamente realiza el etiquetado de la imagen binaria, y todo ello en un único barrido de la imagen. Actualmente pasa por ser uno de los algoritmos más rápidos de etiquetado, y es nuestro algoritmo de referencia.

Supondremos que, en la imagen binaria *B(x,y)*, las regiones tienen valor 1 y el fondo 0. Para evitar que el algoritmo tenga que comprobar que no se accede a posiciones fuera de los límites de la imagen, se supondrá que no existe ninguna región en contacto con estos límites o se pondrán directamente a 0 los píxeles en los bordes de la imagen.

El algoritmo puede explicarse en función de las diferentes decisiones que se adoptan según las propiedades de los diferentes puntos de una región. Al iniciarse el barrido de una fila desactivamos una bandera, *etiq_activa=0,* que nos servirá para indicar si una región está siendo visitada. Así, durante el barrido de una fila de la imagen binaria *B(x,y)*:

1. Encontramos por primera vez un punto P_e de una región (figura 4.12.a). En este momento creamos una nueva etiqueta *netiq* con la que será etiquetada la región y activamos *etiq_activa=netiq*. El punto P_e es un punto del contorno exterior de la región *netiq*. Mediante una llamada a la función ***SEGUIDORCONTORNO()*** se creará una lista con los puntos de este contorno y los etiquetará con el valor *etiq_activa* en la imagen de etiquetas *E* (figura 4.12.b). Esta función, con independencia de si el contorno es externo o interno, marca en *E* con un valor bandera, por ejemplo -1, todos los 8-vecinos del contorno que no pertenecen a la región. El marcado prevendrá de que un contorno interno se recorra más de una vez.
2. Estando la bandera *etiq_activa* activada, *etiq_activa=netiq*, encontramos un punto *B(x,y)* a 1: entonces *E(x,y)= etiq_activa* (figura 4.12.d).
3. Estando la bandera *etiq_activa* activada, *etiq_activa=netiq*, encontramos un punto *B(x,y)* a 0:
   - Si su etiqueta es -1 (figura 4.12.c) implica que es un punto vecino de los contornos ya recorridos de la región, externo o interno. No se hace nada. La bandera es desactivada, *etiq_activa*=0.
   - Si su etiqueta es 0 (figura 4.12.d), significa que es el primer punto encontrado de uno de los posibles agujeros de la región. El punto inmediatamente superior P_i=*(x-1,y)* por fuerza es un punto de un contorno interior. Una llamada a ***SEGUIDORCONTORNO()*** creará una lista con los puntos de este contorno interno y los etiquetará con el valor *etiq_activa* en la imagen de etiquetas *E* (figura 4.12.e).
4. Estando *etiq_activa* desactivada encontramos un punto *B(x,y)* del contorno exterior o interior de la región etiquetado con un valor *netiq*. Por ejemplo, el punto Q_i en figura 4.12.f. Obviamente esto implica que el punto anterior *B(x,y-1)* está a 0. Activaremos la bandera, *etiq_activa=E(x,y)*, que indica que una región está siendo visitada.

El pseudocódigo puede verse en el Algoritmo 4.6. Ya hemos visto que la función ***SEGUIDORCONTORNO()*** realiza varios cometidos:

- Obtiene los contornos externo e internos de una región
- Marca con el valor de etiqueta correspondiente los puntos de contorno
- Marca con una etiqueta bandera -1, los puntos 8-vecinos de los contornos que pertenecen al fondo.

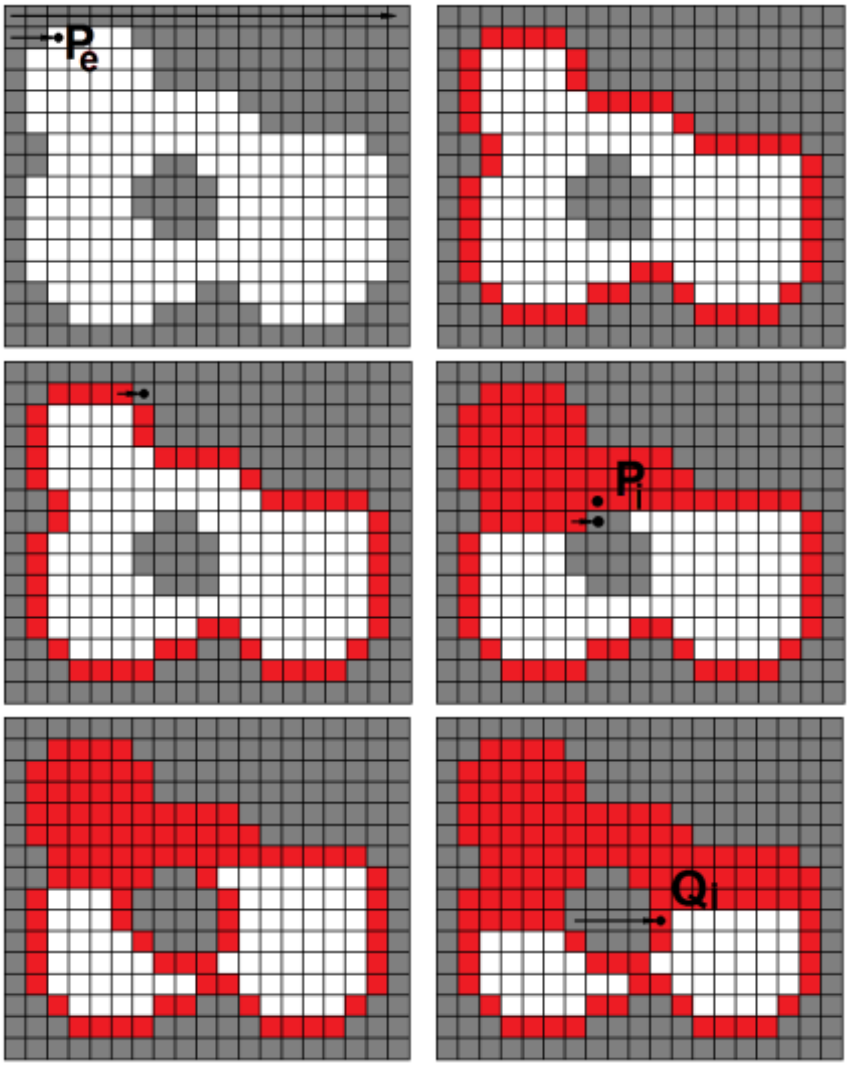


Figura 4.12 Casuística del algoritmo del seguidor de contornos y etiquetado simultáneos

La función ***SEGUIDORCONTORNO()*** debe garantizar que no termina de forma prematura al visitar de nuevo el punto de inicio. La *condición de parada* consiste en salvaguardar: el punto inicial, P_e o P_i, (que se guardará en la última posición del contorno, punto P_{fin}) y su sucesor P_{ini} (que se guardará en la primera posición). La parada ocurre cuando en el trascurso del seguimiento el punto actual P_{act} y su sucesor P_{sig} coinciden respectivamente con P_{fin} y P_{ini}. Además, esta función debe contemplar la presencia de una región monopíxel: se detecta fácilmente sin más que comprobar al inicio que P_{ini} coincide con P_{fin}.

```
→ B:  Imagen binaria de tamaño (filas,columnas).
← E:  Imagen etiquetada de tamaño (filas,columnas)
  C:  Estructura de datos con el contorno exterior y los interiores de cada región
SEGUIDORETIQUETADOSIMULTANEOS(B)
    Inicializar netiq=0, imagen de etiquetas E=0, C=ϕ
    Poner a 0 B(:,1), B(1,:), B(filas,:) y B(:,columnas)   %A 0 los bordes de la imagen
    Para cada x,  x=2,…filas-1
        etiq_activa=0
        Para cada y,  y=2…columnas-1
            Si B(x,y)==1
                Si etiq_activa≠0                    %Etiquetando la región etiq_activa
                    E(x,y)=etiq_activa
                en caso contrario
                    etiq_activa=E(x,y)
                    Si etiq_activa==0                              %Nueva región
                        netiq=netiq+1
                        etiq_activa=netiq
                        Pe=(x,y)                %Punto inicial del contorno exterior
                        ce=SEGUIDORCONTORNO(B,E,etiq_activa,Pe,0)
                        Añadir ce a C(etiq_activa)
                        E(x,y)=etiq_activa
                    Fin
                Fin
            en caso contrario
                Si etiq_activa≠0                %Acabamos de abandonar la región
                    Si E(x,y)==0   %Primer punto encontrado en agujero interno
                        Pi=(x-1,y)              %Punto inicial del contorno interior
                        ci =SEGUIDORCONTORNO(B,E,etiq_activa,Pi,3)
                        Añadir ci a C(etiq_activa)
                    Fin
                    etiq_activa=0
                Fin
            Fin
        Fin
    Fin
    Devolver E,C
```

Algoritmo 4.6: Seguidor de contornos y etiquetado simultáneos.

La función ***SEGUIDORCONTORNO()*** debe decidir para cada punto P_{act} del contorno donde buscar su sucesor P_{sig}. Salvo en los casos particulares de los puntos de entrada a la función, cualquiera que sea la configuración del punto P_{act}, el punto sucesor P_{sig} se hallará buscando en el sentido horario uno de sus 8-vecinos.

```
→ B:          Imagen binaria de tamaño (filas,columnas)
  E:          Imagen etiquetada de tamaño (filas,columnas). Se modifica.
  etiq:       Etiqueta de la región
  Pfin=(x,y): Coordenadas del punto de entrada. Se almacena al final del contorno
  codigo:     Código inicial de búsqueda respecto a Pact (0 exterior, 3 interior)
← c:          Vector con las coordenadas del contorno
SEGUIDORCONTORNO(B,E,etiq,Pfin,codigo)
    Inicializar c=0
    contador=1
    (Psig,codigo)=SIGUIENTEPUNTO(B,E,Pfin,codigo)
    c(contador)=Psig                                  %Primer punto del contorno
    Si Psig≠ Pfin                                     %El punto no es un punto aislado
        seguir=1
        Mientras seguir==1                            %Búsqueda en los 7 vecinos
            E(x,y)=etiq                               %Etiquetamos los puntos de contorno
            codigo=MOD(codigo+6,8)
            Psnt=Psig
            (Psig,codigo)=SIGUIENTEPUNTO(B,E,Pant,codigo)
            Si Psig==c(1) AND Pant== Pfin
                seguir=0                              %Se cumple condición de parada
            en caso contrario
                contador=contador+1
                c(contador)=Psig
            Fin
        Fin
    Fin
    Devolver c
```

Algoritmo 4.6: Seguidor de contornos y etiquetado simultáneos. Función ***SEGUIDORCONTORNO*** ()

5	6	7
4	**P**	0
3	2	1

Figura 4.13. Código de vecindad de Moore del punto P

Para decidir por cual de esos 8-vecinos iniciamos la búsqueda, se utilizan unos códigos, habitualmente denominados como *vecindad de Moore* o *códigos de Freeman* según autores (figura 4.13).

El código correspondiente a los puntos iniciales de los contornos es diferente según se trate del punto inicial de un contorno exterior o interior.

```
→ B:          Imagen binaria de tamaño (filas,columnas)
  E:          Imagen etiquetada de tamaño (filas,columnas). Se modifica.
  Pact=(x,y): Coordenadas del punto actual en el contorno
  codigo:     Código inicial de búsqueda respecto a Pact
← Psig=(x,y)  Coordenadas del punto sucesor Psig si existe, o Psig=Pact si no existe
  codigo      Código final con la posición, si existe, de Psig respecto a Pact
SIGUIENTEPUNTO (B,E,Pact,codigo)
    exito=0
    contador=1
    Mientras exito==0 AND contador<8                    %Búsqueda en los 7 vecinos
        Psig=(x,y)=Pact+VECINO(codigo)                  %Nuevas coordenadas de búsqueda
        Si B(x,y)==0                                    %El punto no pertenece a la región
            E(x,y)=-1                                   %Marcamos el píxel
            codigo=MOD(codigo+1,8)                      %Siguiente posición según agujas del reloj
        en caso contrario
            exito=1
        Fin
        contador=contador+1
    Fin
    Si exito==0
        Psig=Pact                                       %No se ha encontrado ningún vecino: punto aislado
    Fin
    Devolver Psig,codigo
```

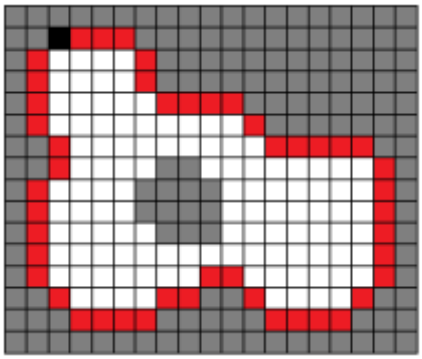
Algoritmo 4.6: Seguidor de contornos y etiquetado simultáneos. Función ***SIGUIENTEPUNTO***()

Si el punto de entrada P_e corresponde a un contorno exterior, este punto tiene sus 8-vecinos de la fila situada por encima igual a 0. Por tanto, basta con comenzar la búsqueda en el píxel situado al Este, es decir, *codigo=0*.

Si el punto de entrada P_i corresponde a un contorno interior, este punto está situado al Norte del primer punto del agujero de la región, y el píxel que precede a este último es un punto de la región, 8-vecino de P_i situado al Suroeste, es decir *codigo*=3.

Para el resto de casos, todo lo que necesitamos es saber el código del punto P_{act} respecto a su antecesor P_{ant}. Basta calcular el nuevo código de posición mediante la operación *codigo=**MOD***(*codigo*+6,8), donde ***MOD()*** es la función resto de la parte entera.

Veámoslo con un ejemplo: Supongamos que P_{ant} es el punto situado al Suroeste de P_{act}. Por tanto, el código con el que se accedió al punto P_{act} fue el 7, pues P_{act} está al Noreste de P_{ant}. Dado que P_{act} se ha obtenido mediante una búsqueda según las agujas del reloj centrada en P_{ant}, el punto situado al oeste de P_{act} es un punto que se ha visitado y no se ha seleccionado, es decir, no pertenece al contorno y, por tanto, podemos obviarle en la búsqueda. Ya centrada la búsqueda entorno a P_{act} y siguiendo el sentido de las agujas del reloj,



(2,3),0,0,0,1,2,1,0,0,0,1,1,0,0,0,0,1,2,2,2,2,2,…

Figura 4.14. Representación del contorno utilizando los códigos de Freeman. En primer lugar figuran las coordenadas del píxel de inicio del contorno que aparece marcado en negro en la figura. A continuación se apunta cada píxel de contorno con un único dígito entre 0 y 7 dependiendo de la posición que ocupen respecto al píxel de contorno anterior.

el siguiente punto sobre el que no tenemos ninguna información es el punto situado al Noroeste, es decir, el punto con código ***MOD***(*7*+6,8)=5 respecto a P_{act}.

La función ***SIGUIENTEPUNTO()*** realiza la búsqueda del siguiente punto de contorno en la 8-vecindad de P_{act}, partiendo del vecino situado en la posición determinada por *codigo* y siguiendo las manecillas del reloj. Si tras completar el ciclo de 7 visitas no encontrásemos ningún punto, significaría que el punto P_{act}, por fuerza P_e, ¡es un *punto aislado*!. Incidimos de nuevo en que el 8º vecino a partir de *codigo* siempre será un punto que no pertenece al contorno la región.

A partir de esta posición inicial se visitan los vecinos de P_{act} tal que:

- Si el punto P_{sig} no pertenece a la región, se *marca* con un valor de etiqueta no válido, por ejemplo -1, y se pasa al siguiente vecino según el movimiento de las agujas del reloj.
- Si el punto P_{sig} pertenece a la región se devuelven sus coordenadas y el nuevo valor *codigo*.

A la hora de almacenar los puntos de contorno tenemos la opción de guardar tan solo las coordenadas del punto inicial del contorno y, a continuación, los códigos de acceso de la vecindad de Moore (figura 4.14), aunque esta codificación encadenada es más conocida en este ámbito como códigos de Freeman (Freeman, 1961). Existen numerosos algoritmos que utilizando estos u otros tipos de códigos de cadena, no solo ahorran espacio en memoria sino que sirven para el reconocimiento de formas (Sánchez, 2006) y, en especial, el de caracteres. Sin embargo, su uso en aplicaciones industriales podemos considerarlo totalmente residual, dada la fuerte dependencia que el código de cadena tiene respecto del punto inicial y su no invarianza a la escala.

4.5. EXTRACCIÓN DE CARACTERÍSTICAS EN IMÁGENES BINARIAS

En las aplicaciones industriales de visión artificial se tiene siempre un conocimiento a priori del objeto u objetos que van a aparecer ante la cámara. Saber de antemano cómo son los objetos permitirá establecer propiedades que sean especialmente discriminantes entre estos objetos. Entre estas características tenemos el área que ocupan en la imagen, número de agujeros, momentos de inercia, etc. Cuantificando estas propiedades se podrá llegar a diferenciar unos objetos de otros sin más que comparar valores numéricos. Esta cuantificación de las propiedades de los objetos que aparecen en la imagen es lo que se conoce como extracción de características.

La extracción de características consiste en el cálculo de propiedades globales de las regiones que aparecen en la imagen. Por tanto, se pasa de trabajar desde propiedades de píxel a un nivel de propiedades de regiones que inicialmente no es explícito en la imagen. Esto tiene muchas ventajas ya que, por un lado, las regiones serán en principio mucho menores en número y, por otro lado, tienen muchas más propiedades que un píxel. Un píxel solo tiene como características su posición y su nivel de gris mientras que una región, aunque sea de una imagen binaria, tiene un contenido semántico mucho mayor pues se puede establecer su área, perímetro, elongación o circularidad, orientación, número de agujeros, momentos de inercia, etc. El análisis posterior sobre regiones es mucho más fiable que si se lleva a nivel de píxel.

A la hora de elegir las características más adecuadas para describir una región es deseable que estas reúnan una serie de propiedades tales como:

- *Capacidad discriminante*

  Los valores numéricos asociados a una característica deben ser significativamente distintos para poder distinguir unos objetos de otros.

- *Estabilidad*

  La característica o características elegidas deben presentar valores numéricos *similares* para una misma clase de objetos. Si el objeto que debe reconocerse puede aparecer en distintas posiciones en la imagen las características elegidas deberán ser invariantes ante traslación y rotación. Si el objeto tiene distintos tamaños deberán ser invariantes a cambios de escala.

- *Fácilmente evaluables*

  El cálculo de las características no debe requerir de una carga computacional exagerada para que puedan obtenerse dentro del tiempo disponible en la aplicación.

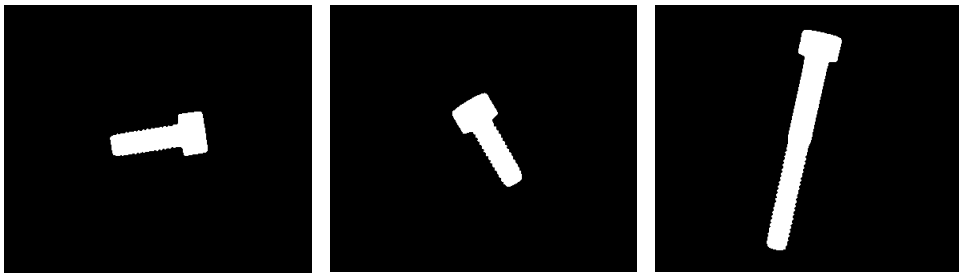


Figura 4.15. Supongamos que tenemos una aplicación de clasificación de tornillos. Sabemos que en la línea de clasificación van a aparecer únicamente dos tipos de tornillos: unos pequeños con un área entorno a los 4000 píxeles en la imagen y otros grandes, con un área que supera los 8000 píxeles en la imagen. En estas circunstancias, el área o tamaño del objeto en la imagen es una característica discriminante que se puede utilizar para distinguir unos de otros. Por ejemplo, el área obtenida sobre cada una de estas imágenes ha sido 4015, 3995 y 8403 respectivamente. La elección de esta característica para diferenciarlos es correcta pues los valores del área no dejan lugar a dudas sobre la pertenencia de un tornillo a un conjunto u otro (capacidad discriminante), es estable ante los cambios de posición que pueden presentar los tornillos ante la cámara y además es fácilmente evaluable.

Por simplicidad y eficiencia interesa trabajar con el mínimo número de características. Si dos características están relacionadas, alguna de ellas no aportará gran cosa y lo mejor será eliminarla. Cuando el número de objetos diferentes que pueden aparecer en la imagen es reducido, como es el caso de la mayoría de las aplicaciones de inspección industrial (pieza grande o pequeña, envase con tapón o sin él, pieza con o sin agujero, artículo con mancha o sin ella, etc.) no es difícil establecer a priori una o dos características que representen de forma inequívoca cada clase de objeto. Sin embargo, en aplicaciones más complejas con gran número de patrones, por ejemplo de reconocimiento de caracteres, elegir las características más adecuadas para diseñar el clasificador no es una tarea tan sencilla.

Algunas de las características más utilizadas para reconocer y localizar objetos en imágenes son:

- Tamaño
- Posición
- Orientación
- Perímetro y circularidad
- Polígonos envolventes
- Número de agujeros y número de Euler

Consideraremos a partir de ahora, sin pérdida de generalidad, que trabajamos con una imagen con una única región etiquetada como 1. La extensión a calcular las características de las regiones de una imagen etiquetada con varias regiones es muy simple.

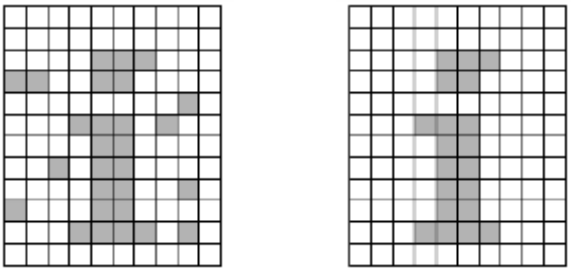


Figura 4.16: A veces las imágenes binarias presentan ruido que origina regiones dispersas de pequeño tamaño comparado con las regiones de interés. Empleando un filtro de tamaño es posible eliminar todos los componentes que no superen un determinado umbral de tamaño.

4.5.1 Tamaño

Suponiendo píxeles cuadrados, que es actualmente lo estándar, el *área* de una región vendrá dada por el número de píxeles que la constituyen.

Así, el área se obtiene fácilmente como:

$$Area = \sum_{x=1}^{filas} \sum_{y=1}^{columnas} B(x,y) \tag{4.1}$$

A este parámetro también se le denomina momento de orden cero. El tamaño de un objeto en la imagen es invariante ante la rotación y traslación, es decir, no depende de la posición en la que aparezca el objeto en la imagen (figura 4.16).

Como hemos visto con anterioridad, algunos algoritmos de etiquetado proporcionan de forma directa el área de sus regiones.

4.5.2 Posición

La posición de un objeto en la imagen normalmente se calcula determinando la posición del *centro de gravedad* de los píxeles que integran el objeto, en los que estos se consideran de masa unitaria. El centro de gravedad es poco sensible al ruido y resulta muy útil en la aprehensión de objetos con manipuladores (figura 4.17).

Insistiendo en que suponemos imágenes con una única región, el centro de gravedad (x_g, y_g) de la región se obtiene a partir de sus coordenadas:

$$x_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} x \cdot B(x,y)}{Area} \qquad y_g = \frac{\sum_{x=1}^{f} \sum_{y=1}^{c} y \cdot B(x,y)}{Area} \tag{4.2}$$

En definitiva, son los momentos de primer orden divididos por el área.

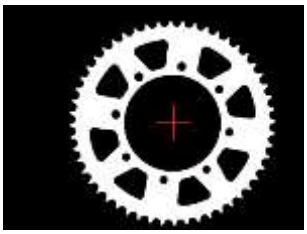


Figura 4.17. El centro de gravedad es una de las propiedades más útiles y robustas a la hora de guiar manipuladores para capturar objetos con simetría radial.

4.5.3 Orientación

La orientación de un objeto puede determinarse mediante el cálculo del eje respecto al cual el momento de inercia es mínimo. Utilizando la definición que se hace en mecánica, el eje de inercia mínimo de una región será la recta $Ax + By + C = 0$ tal que la suma de las distancias al cuadrado entre los píxeles del objeto y dicha línea es mínima. Téngase en cuenta que el vector $(A, B) = (\cos\theta\,, \operatorname{sen}\theta)$ es un vector perpendicular a la recta, siendo θ el ángulo que forma la perpendicular desde el origen a la recta con el eje *x* (figura 4.18).

A efectos didácticos, y dado su gran aplicación en múltiples problemas, vamos a derivar la obtención de la ecuación que gobierna esta recta.

Dado que el centro de gravedad (x_g, y_g) es un punto que pertenece a los ejes principales de inercia, debe cumplirse:

$$Ax_g + By_g + C = 0 \tag{4.3}$$

y sustituyendo el valor de *C*:

$$A(x - x_g) + B(y - y_g) = 0 \tag{4.4}$$

El problema puede entonces formularse de forma sencilla trasladando la región al origen según (x_g, y_g) con lo que el eje principal de inercia en las nuevas coordenadas $x' = (x - x_g)$ e $y' = (y - y_g)$ pasará por el origen y tendrá por ecuación:

$$Ax' + By' = 0 \tag{4.5}$$

Dado que el vector $(A, B) = (\cos\theta\,, \operatorname{sen}\theta)$ cumple $A^2 + B^2 = 1$, la distancia al cuadrado d_i^2 de cualquier punto (x_i', y_i') del objeto a la recta vendrá dada por:

$$d_i^2 = (Ax_i' + By_i')^2 \tag{4.6}$$

por lo que la suma de todas las distancias al cuadrado $\sum_{i=1}^{N} d_i^2$ vendrá dada por:

$$\sum_{i=1}^{N}(Ax_i' + By_i')^2 = A^2 \sum_{i=1}^{N} x_i'^2 + B^2 \sum_{i=1}^{N} y_i'^2 + 2AB \sum_{i=1}^{N} x_i' y_i' \tag{4.7}$$

donde *N* es el número de píxeles de la región, es decir, el área.

Es fácil identificar en los sumatorios de (4.7), deshaciendo la traslación previamente efectuada al origen, los tres momentos de segundo orden de una región I_{xx}, I_{yy} e I_{xy}:

$$I_{xx} = \sum_{x=1}^{f}\sum_{j=1}^{c}(x-x_g)^2 \cdot B(x,y) = \sum_{i=1}^{N}(x_i-x_g)^2$$
$$I_{xy} = \sum_{x=1}^{f}\sum_{j=1}^{c}(x-x_g)\cdot(y-y_g)\cdot B(x,y) = \sum_{i=1}^{N}(x_i-x_g)(y_i-y_g) \quad (4.8)$$
$$I_{yy} = \sum_{x=1}^{f}\sum_{j=1}^{c}(y-y_g)^2 \cdot B(x,y) = \sum_{i=1}^{N}(y_i-y_g)^2$$

Por tanto, sustituyendo (A,B) por $(\cos\theta\,,\mathrm{sen}\,\theta)$, podemos rescribir (4.7) como:

$$\textstyle\sum_{i=1}^{N} d_i^2 = \cos^2\theta \cdot I_{xx} + \mathrm{sen}^2\theta \cdot I_{yy} + 2\cos\theta\mathrm{sen}\theta \cdot I_{xy} \quad (4.9)$$

Puesto que lo que deseamos es encontrar la recta que minimiza el sumatorio de las distancias al cuadrado, derivaremos (4.9) respecto al parámetro θ e igualaremos a 0:

$$\frac{d}{d\theta}\sum_{i=1}^{N} d_i^2 = 0 \quad (4.10)$$
$$-2\cos\theta\mathrm{sen}\theta \cdot I_{xx} + 2\cos\theta\mathrm{sen}\theta \cdot I_{yy} + 2(\cos^2\theta - \mathrm{sen}^2\theta)\cdot I_{xy} = 0$$

Dado que por trigonometría sabemos que:

$$(\cos^2\theta - \mathrm{sen}^2\theta) = \cos(2\theta)$$
$$2\cos\theta\mathrm{sen}\theta = \mathrm{sen}(2\theta)$$

podemos reformular (4.10) como:

$$\mathrm{sen}(2\theta)\cdot(I_{yy}-I_{xx}) + 2\cos(2\theta)\cdot I_{xy} = 0$$

Finalmente despejando[1],

$$\tan(2\theta) = \frac{2I_{xy}}{I_{xx}-I_{yy}} \quad (4.11)$$

Obviamente, las soluciones $\theta_1 = \theta$ y $\theta_2 = \theta_1 + \pi/2$ son ambas solución al problema, por lo que no podemos garantizar a cual de los dos ejes principales de inercia corresponde el ángulo θ.

[1] Es recomendable calcular el ángulo como `tita=0.5*atan2(2*Ixy,Ixx-Iyy)` para evitar divisiones por cero.

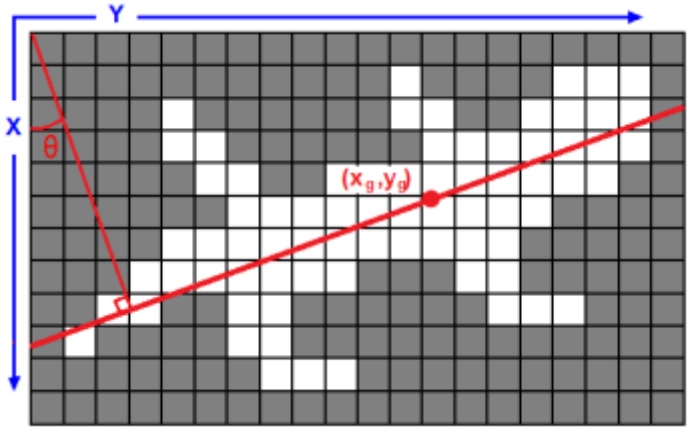


Figura 4.18. El ángulo θ se obtiene fácilmente tras el cálculo de los momentos de segundo orden I_{xx}, I_{yy} e I_{xy}. Los ejes de inercia pasarán por el centro de gravedad del objeto (x_g, y_g).

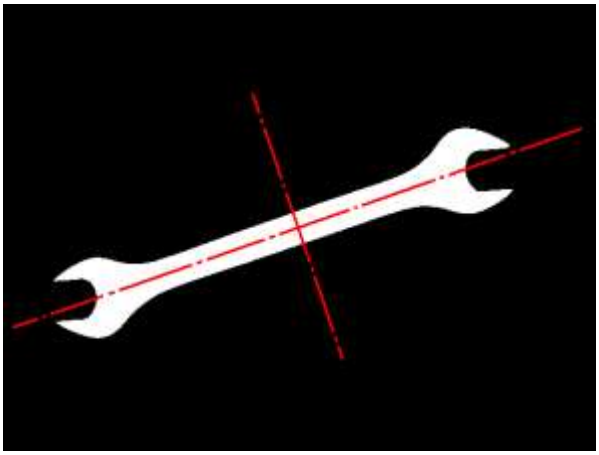


Figura 4.19**.** Muchas aplicaciones industriales requieren conocer la orientación del objeto para su aprehensión. La garra de un robot precisa no solo el centro de gravedad de la pieza sino también debe conocer el giro necesario para que los dedos cierren a lo largo del eje de inercia de momento máximo para tener éxito en la captura.

Para deshacer la indeterminación simplemente basta con sustituir θ_k, $k = 1$ o 2 en (4.9) y elegir el ángulo optimo θ_o que nos dé el menor valor:

$$\theta_o = argmin_{k=1,2}(\cos^2\theta_k I_{xx} + \text{sen}^2\theta_k I_{yy} + 2\cos\theta_k \text{sen}\theta_k I_{xy}) \qquad (4.12)$$

En definitiva, el eje de inercia buscado es:

$$\cos\theta_o x + \text{sen}\theta_o y - (\cos\theta_o x_g + \text{sen}\theta_o y_g) = 0 \qquad (4.13)$$

En el capítulo 8 dedicado a la estimación de parámetros presentaremos otra alternativa al cálculo de esta recta.

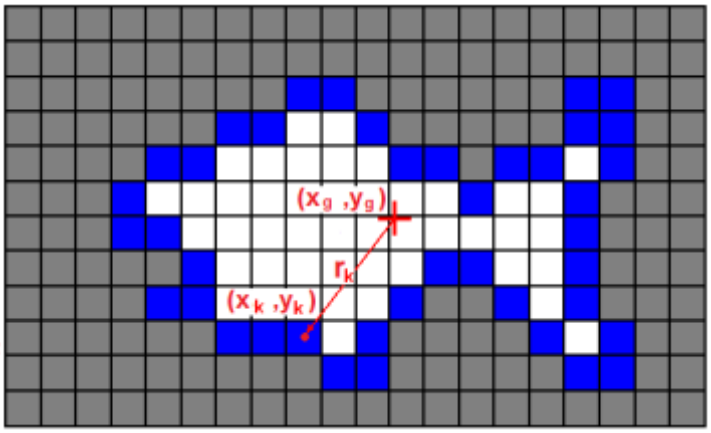


Figura 4.20. La circularidad se puede evaluar calculando la varianza de las distancias de los píxeles de contorno al centro de gravedad.

4.5.4 Perímetro y circularidad

La longitud del perímetro se obtiene simplemente sumando 1 cada vez que la secuencia de píxeles del contorno avanza en vertical u horizontal y sumando $\sqrt{2}$ cuando la secuencia avanza en diagonal.

A partir del contorno no solo disponemos del perímetro del objeto si no que también podemos cuantificar la forma que presenta el objeto de la imagen combinando el valor del perímetro con el del área. Esta nueva característica nos da una idea de su compacidad o circularidad. Supongamos que nuestro objeto tiene un área A y un perímetro P. El radio r correspondiente a una circunferencia con ese perímetro $P = 2\pi r$ será, por tanto, $r = P/2\pi$. Entonces, el área que presenta dicha circunferencia de perímetro P vendrá dado por:

$$A = \pi r R^2 = \pi \left(\frac{P}{2\pi}\right)^2 = \frac{P^2}{4\pi} \tag{4.14}$$

Para cuantificar la circularidad de un objeto compararemos el área de la circunferencia A_c con el área de dicho objeto:

$$circularidad = \frac{A}{A_c} = \frac{4\pi A}{P^2} \tag{4.15}$$

El valor de la circularidad oscila entre 0 y 1. Sobre objetos redondeados toma valores próximos a 1 y para objetos irregulares o alargados la circularidad tiende al cero.

Obviamente, si los objetos presentan agujeros, el área a considerar en la fórmula anterior no será el del objeto real sino el del objeto sumando el área de sus agujeros.

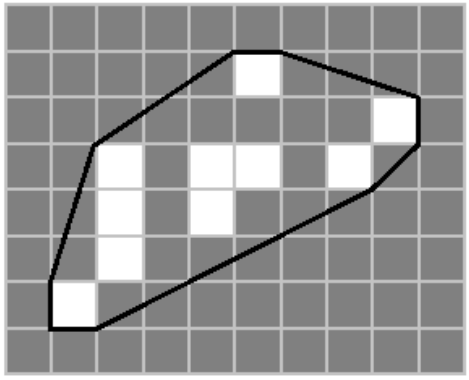


Figura 4.21 Envolvente convexa

Otra medida de circularidad consiste en obtener la media μ y la desviación estándar σ de las distancias r_k del centro de gravedad a cada uno de los k píxeles que forman el contorno de la región C formado por P píxeles:

$$r_k = \left\|(x_g, y_g) - (x_k, y_k)\right\|, \quad (x_k, y_k) \in \text{contorno } C$$
$$\mu = \frac{\sum_{k \epsilon \mathrm{P}} \; r_k}{P} \tag{4.16}$$
$$\sigma^2 = \frac{\sum_{k \epsilon \mathrm{P}} \; (r_k - \mu)^2}{P}$$

La relación μ/σ crece, teóricamente hasta el infinito, cuanto más circular es una figura, siendo independiente del área, la orientación y la escala.

Si hay agujeros en la región también es preciso eliminarles para hacer el cálculo de la circularidad con este nuevo planteamiento, pues desplazan la posición de μ.

4.5.5 Envolvente convexa y MBR.

La envolvente convexa (*convex hull*) de un conjunto de píxeles C es la figura poligonal convexa más pequeña que contiene al conjunto. Intuitivamente podemos imaginar la envolvente convexa de la siguiente forma. Supongamos que sobre cada píxel del conjunto clavamos una punta (sin hundirla hasta la cabeza). La envolvente convexa sería la forma que adoptaría una goma elástica que encerrase al conjunto de puntas y que correspondería a un polígono convexo cuyos vértices son algunos de los píxeles del contorno de la región (figura 4.21).

El cálculo de la envolvente convexa ha sido muy estudiado en gráficos por ordenador y existen variados algoritmos para su cálculo que no veremos por exceder los límites de este libro.

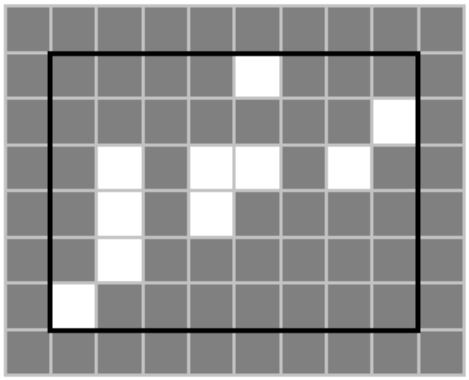


Figura 4.22 Rectángulo envolvente de tamaño mínimo

Otra característica sencilla de calcular y que da una idea de la extensión y posición de un conjunto de píxeles C en la imagen es el mínimo rectángulo contenedor *MBR (mínimum bounding rectangle*), cuyos ejes son paralelos a los de la imagen (figura 4.22). El MBR tendrá como coordenadas de su esquina superior izquierda a $(\min(x),\min(y))$ y de su esquina inferior derecha a $(\max(x),\max(y))$, con $(x,y) \in C$.

4.5.6 Número de agujeros y número de Euler.

Si los objetos que pueden aparecer en la imagen presentan distinto número de agujeros, resulta muy eficaz utilizar este parámetro para discriminar entre unos y otros.

Para contabilizar el número de agujeros lo más sencillo es invertir la imagen y etiquetar a continuación esa imagen inversa. En la inversión, los agujeros que estaban en negro se transformarán a blanco junto con el fondo de la imagen. Luego, en el etiquetado, se asignarán etiquetas a todos los objetos que aparecen la imagen inversa. Los nuevos objetos serán ahora los agujeros. Además se etiquetará también el fondo de la escena como un objeto al estar en blanco (figura 4.23). Por tanto, se tiene que el número de agujeros del objeto será inferior en una unidad al número de etiquetas obtenidas:

$$numAgujeros = numEtiquetasImgInversa - 1 \qquad (4.17)$$

Si tenemos imágenes con varios objetos es importante indicar que para contabilizar el número de agujeros de un objeto la imagen a invertir debe contener solo ese objeto. Si no, contabilizaremos el número total de agujeros del total de objetos que hay en la imagen, no solo los del objeto de interés. Para evitar este problema, a partir del etiquetado de la imagen original será preciso para cada etiqueta crear una imagen intermedia donde solo figure cada objeto y, sobre cada una de ellas, se realizará el análisis del número de agujeros.

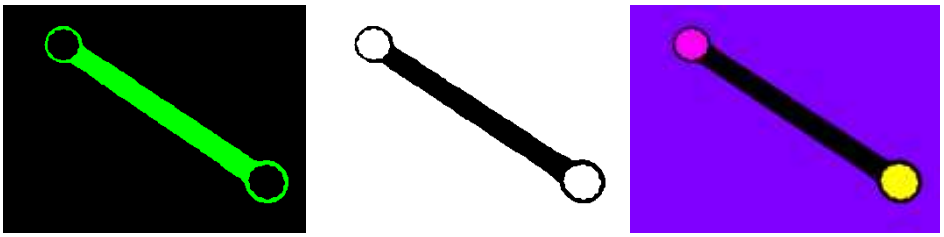


Figura 4.23. a) Imagen resultante del etiquetado de una imagen formada por una única región y, por tanto, solo tenemos una única etiqueta correspondiente al único objeto presente en la imagen. b) Inversa de la imagen binaria original. c) Imagen resultante del etiquetado de la imagen b. En este caso aparecen tres etiquetas correspondientes a las tres regiones conexas de c: los dos agujeros de la herramienta y la región correspondiente al fondo.

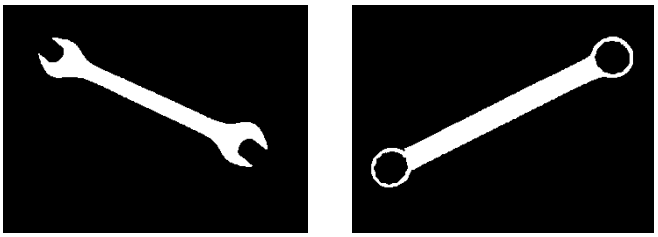


Figura 4.24. La presencia de agujeros en un objeto puede resultar muy útil para diferenciarlo de otros. Por ejemplo, en una aplicación donde se presenten piezas con similar área y forma, pero con agujeros en unas y en otras no, se pueden diferenciar con total garantía sin más que calcular el número de agujeros en ellas.

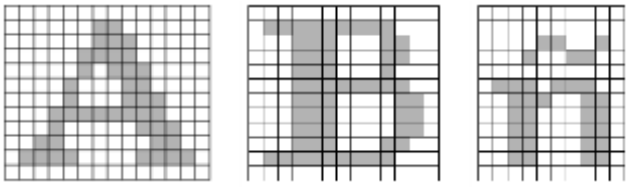


Figura 4.25. a) El número de Euler para A = 1 -1 = 0. b) El número de Euler para B = 1 -2 = -1. c) El número de Euler para ñ = 2-0 = 2

En determinadas aplicaciones un mismo objeto puede estar formado por dos o más regiones no conexas. Por ejemplo, en reconocimiento de caracteres la *ñ*, la *j*, la *i*, … son objetos formados por dos componentes no conexas (figura 4.25). En estas aplicaciones, resulta útil calcular el valor que toma el *Número de Euler*. El Número de Euler se define como el número de componentes menos el número de agujeros.

$$\textit{númeroEuler} = \textit{númeroComponentes} - \textit{númeroAgujeros} \qquad (4.18)$$

El número de agujeros y el número de Euler es una característica invariante a la traslación, rotación y escala.

En cualquier caso, son pocas las aplicaciones industriales en las que este análisis basado en agujeros es factible o útil.

`>> B = im2bw(I,umbral);` Binariza la imagen `I` con el umbral especificado.
`>> umbralOpt=graythresh(I)` Halla el umbral [0,1] de binarización mediante el método de Otsu.
`>> CC=bwconncomp(B)` Etiqueta los objetos blanco y guarda en la estructura `CC`.
`>> L = labelmatrix(CC);` Crea la imagen de etiquetas a partir de la estructura `CC`.
`>> BW2 = bwboundaries(BW1)` Devuelve las coordenadas de los bordes de todos los objetos y sus agujeros
`>> BW2 = imfill(BW1,'holes')` Rellena a 1 los agujeros de las regiones de la imagen `BW1`, generando la imagen `BW2`.
`>> reg = regionprops (L,'EulerNumber');` Calcula las características de los objetos en la imagen. Se puede especificar también *'Area'*, *'Centroid'*, *'Orientation'*, *'Eccentricity'*, …
`>> numEuler = bweuler(BW)` Número de Euler de una imagen binaria `BW`.
`>> BW2 = bwperim(BW1)` Devuelve una imagen binaria con los píxeles del contorno de los objetos.

Tabla 4.1. Algunas funciones de MATLAB para trabajar con imágenes binarias.

Figura 4.26. Las cajas aparecen en las imágenes con una posición y orientación aleatoria.

4.6 BINARIZACIÓN Y EXTRACCION DE CARACTERÍSTICAS CON MATLAB

En este apartado ilustraremos las técnicas para el tratamiento de imágenes binarias empleando el entorno de programación de MATLAB.

Para ello trabajaremos sobre la siguiente aplicación práctica. Supondremos que por una cinta transportadora discurren unas cajas que se diferencian en la referencia impresa que se encuentra en su cara superior.

Consideraremos que solo existen tres tipos de referencias impresas, cuyos códigos alfanuméricos son: HF9HA-1B, HF12.5HA-1B y HF25HA-1B.

Las cajas llegan al punto de embalado con una posición y orientación aleatorias (figura 4.26).

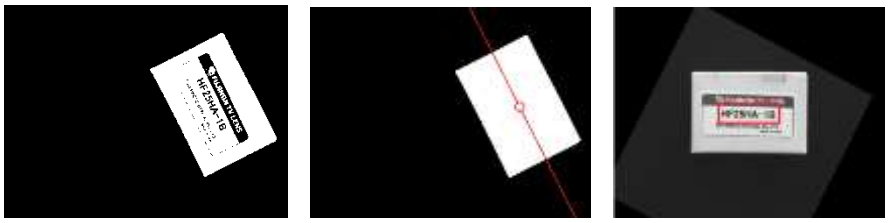


Figura 4.27. a) Imagen binaria. b) Imagen de la caja sin agujeros. Sobre la imagen aparecen sobreimpresionados el centro de gravedad y la orientación. c) Imagen resultante de la rotación donde se ha marcado la ROI.

4.6.1 Centro de gravedad y orientación

En primer lugar determinaremos la posición del centro de gravedad y la orientación de la caja. Para ello, dado el buen contraste que tienen las imágenes de las cajas, binarizaremos la imagen original. Previamente se elegirá el umbral más idóneo para la binarización:

```
>> umbralOpt = graythresh(I);
>> BW = im2bw(I,umbralOpt);
```

Sobre la imagen binaria resultante (figura 4.27.a) se puede proceder a calcular el eje de inercia y el centro de gravedad. No obstante, si queremos ser más precisos, la figura de la caja debería aparecer completa en la imagen sin los agujeros originados por los caracteres y demás elementos impresos. Por tanto, rellenaremos para eliminar los agujeros con la función `imfill()`. Los agujeros son fácilmente eliminables pues son las regiones etiquetadas en la imagen inversa con excepción de la que está en contacto con los límites de la imagen.

```
>> BW2 = imfill(BW,'holes');
```

En la figura 4.27.b puede verse el resultado de esta operación. El paso siguiente será establecer la posición del centro de gravedad y orientación del único objeto que tenemos en la imagen:

```
>> reg = regionprops(BW2,'centroid','orientation','area');
```

Si en la imagen 4.27.a hubiesen aparecido pequeñas regiones correspondientes a ruido, típicamente algún brillo sobre la cinta de transporte, estas pueden fácilmente filtrarse por tamaño. Bastaría escoger la región de la lista `reg` proporcionada por `regionprops()` con el mayor área.

Los parámetros obtenidos se han sobreimpresionado en la figura 4.27.b

4.6.2 Manipulación geométrica

A continuación situaremos la caja en posición horizontal aplicando un giro en sentido contrario a la orientación proporcionada por el eje de inercia:

```
>> R = imrotate(I,-s(1).Orientation);
```

Sobre la caja horizontal podemos ya recortar la región de interés (ROI) que contiene el texto impreso abriendo una ventana de un tamaño prefijado y centrada en el centro de gravedad de la caja.

La posición del centro de gravedad hay que calcularla de nuevo ya que `imrotate` gira respecto al origen. Otra opción, que no modificaría la posición del centro de gravedad, sería realizar la rotación respecto a este calculando la matriz de transformación a partir de la ecuación (3.4) para lo que utilizaríamos funciones `maketform` y `imtransform`.

A continuación, extraeremos de la imagen binarizada original la ROI, que será una ventana cuyas dimensiones hemos prefijado *altoROI x anchoROI*. Para extraer la ROI centramos la ventana en el centroide y pasamos a `imcrop` las coordenadas del vértice superior izquierdo, la anchura y altura.

```
>> CG = s(1).Centroid;
>> ROI = imcrop(BW,[CG(1)-anchoROI/2,  CG(2)-altoROI/2,…
                          anchoROI,  altoROI ];
```

Es importante señalar que MATLAB tanto para las coordenadas de la ventana de `imcrop` como las que proporciona del centroide no son coordenadas matriciales donde la primera componente son las filas y la segunda las columnas. MATLAB en sus funciones gráficas emplea un sistema de coordenadas donde las *x*'s están en la horizontal y las *y*'s crecen hacia abajo.

El paso siguiente será etiquetar la ROI. Como en este caso, los caracteres figuran en negro, es preciso previamente invertir la imagen:

```
>> ROI = ~ROI;
>> CC_ROI = bwconncomp(ROI);
```

Tras la extracción de las componentes conexas se podrán determinar las características individuales de cada carácter que aparece en la imagen:

```
>> reg_ROI = regionprops(CC_ROI,'Area','Centroid');
```

Conocidos los valores de área y centro de gravedad de cada una de las regiones se pueden eliminar las regiones sin interés. Empezaremos quitando las más pequeñas. Prescindiremos de todas las etiquetas con área sensiblemente inferior a 65 píxeles por ser la que presenta el '1' que es el carácter con menor área. Nosotros hemos eliminado aquellas con área por debajo de 40 píxeles.

Este filtro borrará los puntos sueltos que han podido aparecer por una deficiente binarización, el punto decimal y el guión del texto. Estas últimas son regiones con muy pocos píxeles, y son eliminadas.

Se desean eliminar también las regiones que pueden aparecer en la ROI por encima y debajo del texto. Para ello, hemos descartado todas las regiones que presenten su centro de gravedad en las primeras 10 líneas de la ROI o en las 10 últimas.

Figura 4.28. Dependiendo del número de etiquetas que resulten tras el filtrado podemos inferir el texto impreso sin necesidad de hacer un análisis más exhaustivo: 7 etiquetas (HF9HA-1B), 8 etiquetas (HF25HA-1B), 9 etiquetas (HF125HA-1B).

```
function E=filtra_regiones(E,reg,area_min,num)
[fil_E,col_E]=size(E);
for etiq=1:numEtiq
    if (E(etiq).Area<areaMin ||
          E(etiq).Centroid(2)<num ||
          E(etiq).Centroid(2) > filas_E-num)
      [x,y] = find(E==etiq);
      for m=1:length(x)
        E(x(m),y(m))=0;          %borra etiqueta de img E
      end
    end
end
```

Función 1. Filtrado *ad hoc* del ejemplo descrito en la sección 4.5

Estos sencillos filtros son realizados por la función 4.1.

```
>> E = filtra_regiones(E_ROI,reg_ROI,40,10)
```

Tras esta etapa de filtrado por área y posición del centro de gravedad, las imágenes de etiquetas quedarán como aparecen en la figura 4.28. El texto que viene impreso sobre cada caja se puede inferir contabilizando el número de regiones que han pasado el filtro, ya que solo existen tres referencias impresas: HF9HA-1B, HF12.5HA-1B y HF25HA-1B. Si son 7 las etiquetas presentes sería indicativo de que se trata del primer modelo. La detección de 9 etiquetas implicaría la presencia de una caja del segundo modelo y finalmente la contabilización de ocho etiquetas correspondería al tercero (figura 4.28).

Por supuesto, en un caso de coincidencia en el número de regiones, debería realizarse un análisis más detallado de cada una de las regiones correspondientes a las letras. Sin embargo, aquí simplemente hemos pretendido ilustrar como de un problema aparentemente complejo (en nuestro caso parecería que la única alternativa era un reconocimiento de caracteres) pueden encontrarse alternativas simples y muy robustas.

Queremos llamar la atención también sobre el hecho de que la orientación de las letras no es sensiblemente horizontal. Las razones son:

- la binarización puede dar lugar a la obtención de una región no rectangular, influyendo en el cálculo del eje de inercia y de su centroide.
- la etiqueta puede no estar perfectamente colocada, presentando giros y/o desplazamientos respecto a su posición nominal.

En el capítulo 8 veremos una forma más robusta de afrontar el problema.

4.7 CONCLUSIONES

En aplicaciones industriales que requieren la localización o análisis de la silueta de objetos, la imagen de entrada es a menudo simplificada generando una nueva imagen mucho más simple, en la que cada píxel solo puede tener valor 0 (negro) o 1 (blanco). Esto se lleva a cabo mediante un sencillo proceso llamado binarización, que consiste en asignar el valor 0 a todos los píxeles que estén por debajo de un umbral prestablecido y 1 al resto.

La binarización permitirá, en imágenes sencillas y de alto contraste, obtener regiones conexas que corresponden a los objetos y fondo asignando un valor a los píxeles del objeto y otro a los del fondo. Una vez que estas regiones están definidas, se pueden efectuar sobre ellas una gran variedad de medidas tales como área, centro de gravedad, momentos de inercia, circularidad, etc. parámetros que resultan mucho más representativos a la hora de interpretar la imagen que el simple nivel de gris en unas coordenadas cuando se trabaja a nivel de píxel. Es la etapa de *extracción de características*.

Si en la imagen aparecen varios objetos, para cuantificar dichas propiedades a nivel individual será preciso un proceso previo de asignación de una etiqueta a cada uno de los objetos que aparecen en la imagen. Este proceso se conoce como etiquetado de componentes conexas.

Por último, en base a los valores numéricos que presentan las características de cada objeto, se le asignará a este una categoría, que permitirá reconocer los objetos para llevar a cabo desde la manipulación con un robot en una aplicación de guiado hasta discriminar qué objetos de la imagen son aceptables o cuáles son defectuosos si se trata de una aplicación de inspección.