Lección 2: Conectividad

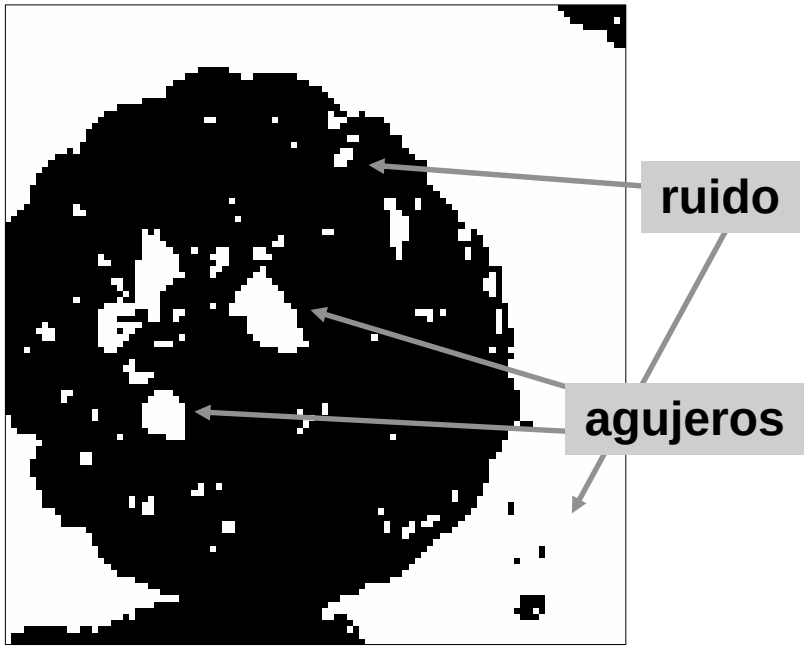
1. Definiciones

2. Algoritmos de etiquetado
   - Recursivo
   - Secuencial

3. Análisis de conectividad
   - RLE
   - Algoritmo secuencial

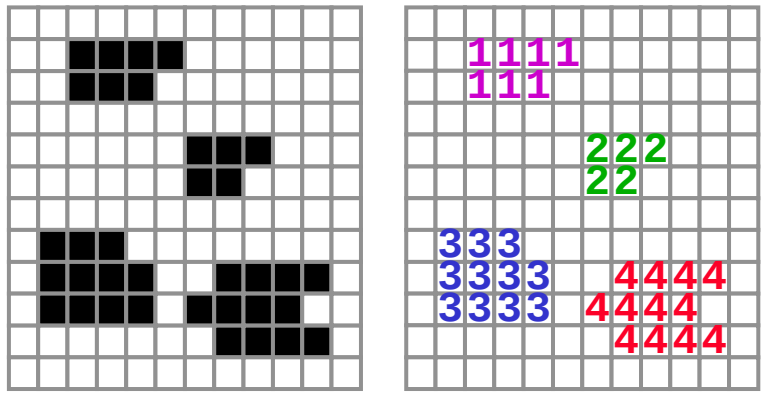
Conectividad

- **Propósito:** separar los objetos de la escena:
  - Del fondo
  - Unos de otros
  - De los agujeros



- Hay ruido de sal y pimienta, que intentamos eliminar

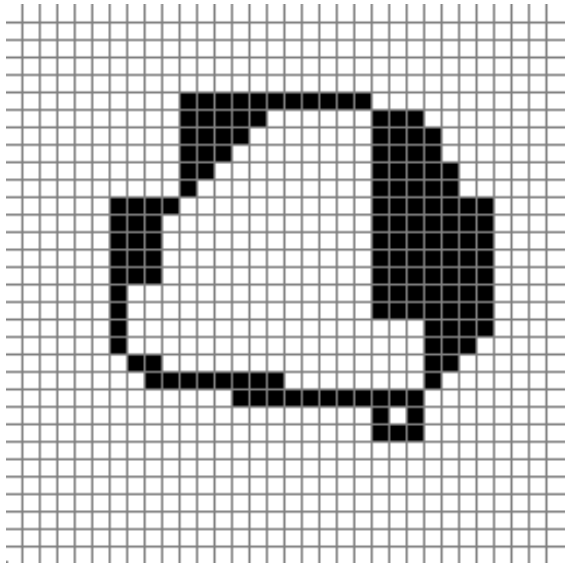
- Se trata de etiquetar cada región de pixels contiguos con un valor diferente



- Trabaja con la imagen binaria
- Genera una imagen coloreada
  - ¿ahorro de espacio?
  - ¿mejora semántica?

Definiciones: conectividad

- **Problema:**



¿cuántos objetos?
¿agujeros, fondo?

- Depende lo que se considere adyacente a un pixel (conectividad).
- También de lo que se considere interior, exterior, etc.

- **Conectividad (vecindad):** proximidad espacial entre pixels de la imagen binaria

4-conectividad

	x_2	
x_3	x_0	x_1
	x_4	

8-conectividad

x_4	x_3	x_2
x_5	x_0	x_1
x_6	x_7	x_8

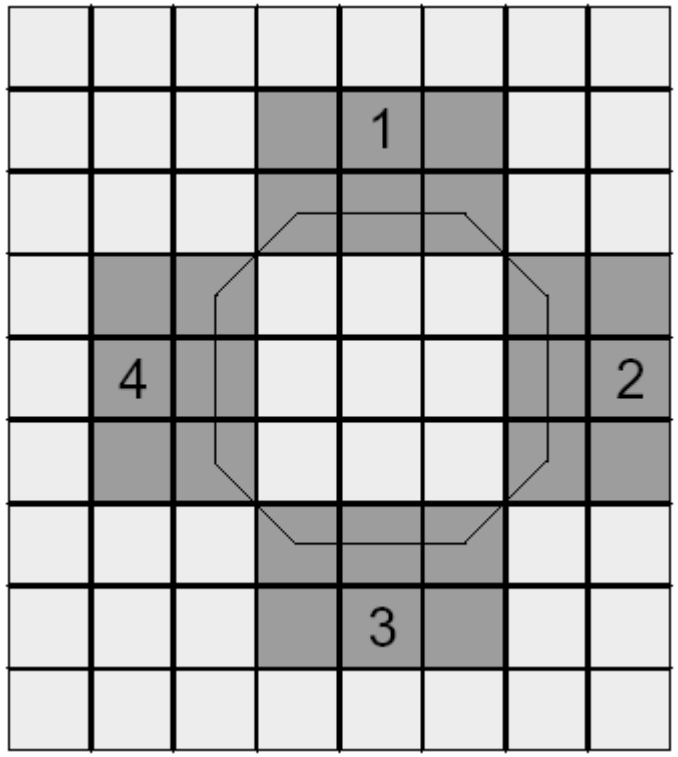
6-conectividad

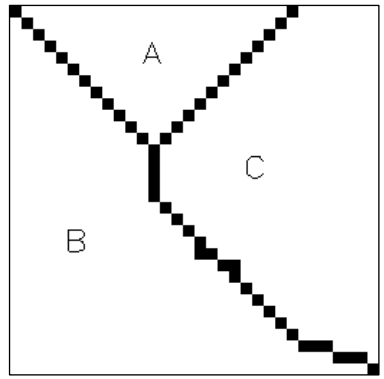
	x_3	x_2
x_4	x_0	x_1
x_5	x_6	

Conectividad

- **Geometría Euclidiana:** una curva cerrada divide el plano en dos regiones no contiguas.
- Dada una imagen digital, una curva cerrada simple (sin cruzamientos), segmenta la imagen en dos regiones conexas disjuntas: interior y exterior.

- **Si se usa 4-conectividad:**
  - cuatro objetos
  - dos regiones disjuntas
  - pero no hay curva de Jordan!
- **Si se usa 8-conectividad:**
  - **un solo objeto**
  - **una curva de Jordan dentro del objeto**
  - **no hay dos regiones disjuntas !!**

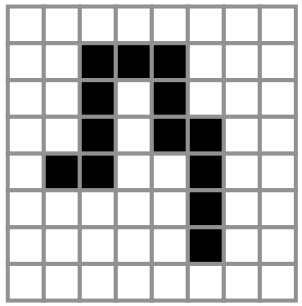




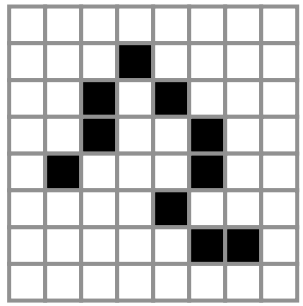
Definiciones

- **Objeto (foreground):** conjunto de 1-pixels (S)
- **Camino:** un camino del pixel $[i_0, j_0]$ al pixel $[i_n, j_n]$ es una secuencia de pixels tal que $[i_k, j_k]$ es vecino de $[i_{k+1}, j_{k+1}]$ para todo $k=0...n-1$

4-conectado



8-conectado



- **Conectividad:** dos pixels p, q $\in$ S están conectados si hay un camino que lleve de p a q cuyos pixels $\in$ S.
- **Es una relación de equivalencia:**
  1. **Reflexividad:** p está conectado a p
  2. **Conmutatividad:** Si p está conectado a q, entonces q está conectado a p
  3. **Transitividad:** Si p está conectado a q, y q está conectado a r, entonces p está conectado a r

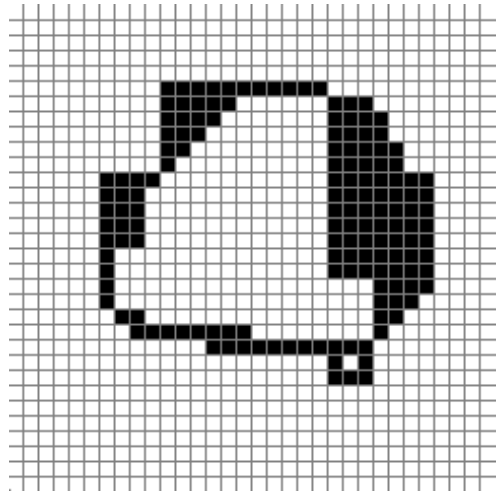
Definiciones

- **Componente conexa (blob):** conjunto de pixels mutuamente conectados.

cada blob corresponderá a un objeto diferente

- **Fondo (background):** conjunto de componentes conectadas del complemento de S ($\overline{S}$) que tienen puntos en el borde de la imagen.
- **Agujeros:** conjunto de componentes conectadas de $\overline{S}$ que NO tienen puntos en el borde de la imagen.

- Volviendo al mismo problema...

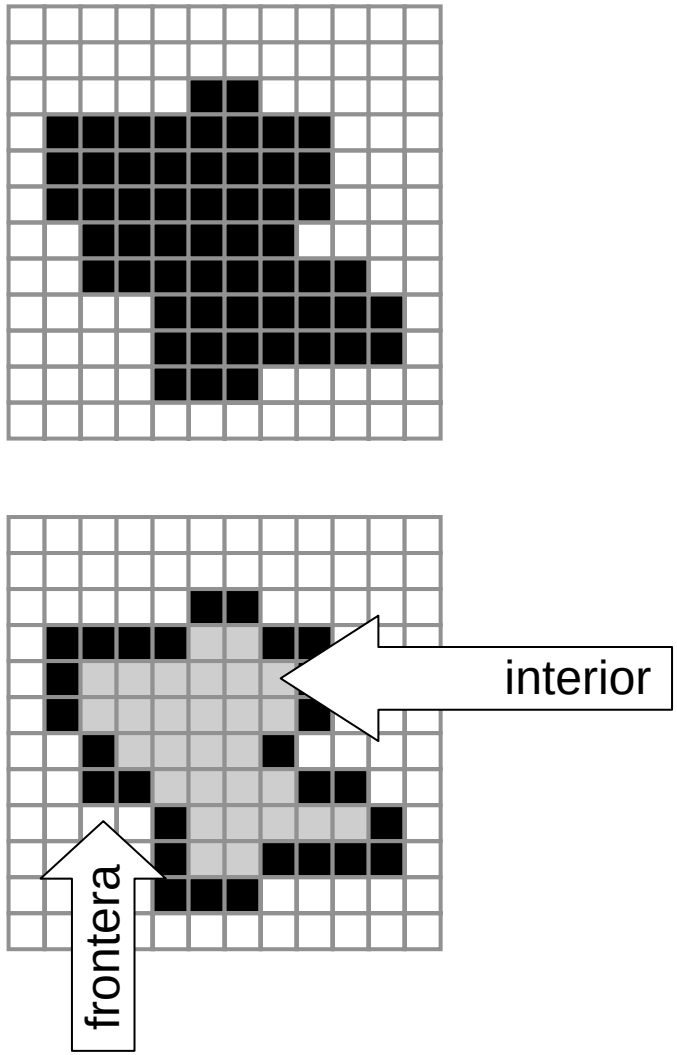


S	$\overline{S}$	Objs	Agujs	
4	4	4	2	!
8	8	1	1	!
				ok
				ok
6	4	?	?	

Para evitar inconsistencias se usa una vecindad para fondo y otra para el objeto, ó ¿6-vecindad para ambos?

Definiciones

- **Frontera (Boundary):** la frontera de S es el conjunto de pixels con vecinos en $\overline{S}$ (suele denotarse como S').
- **Interior:** conjunto de pixels de S que no pertenecen a S' (S - S').
- **Rodear (Surrounds):** una región T rodea a S si cualquier camino desde cualquier punto de S al borde de la imagen, intersecta a T.



Algoritmos etiquetado: recursivo

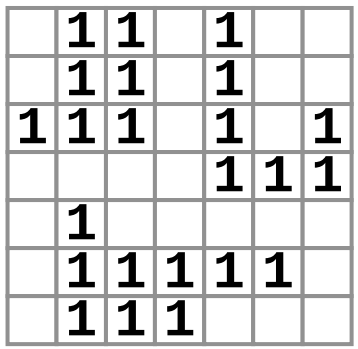
- **Etiquetar:** particionar una imagen binaria en componentes conexas.

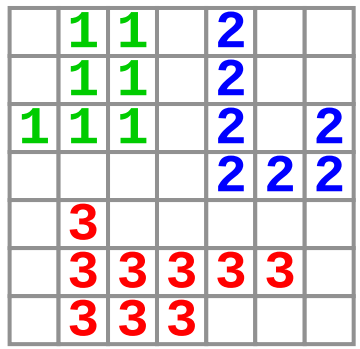
Se espera que las componentes conexas correspondan a objetos o superficies diferentes.

- Se asigna una etiqueta diferente a los pixels de cada componente conexa.
  - **Recursivo:** sencillo pero ineficiente en máquinas secuenciales (es paralelizable).
  - **Secuencial:** requiere dos pasadas, pero sólo dos filas de la imagen (utilizado cuando hay limitaciones de espacio).

Algoritmo recursivo:

1. Buscar un 1-pixel y asignarle una nueva etiqueta L.
2. Recursivamente asignarle la etiqueta L a todos sus vecinos 1-pixels.
3. Si no hay más 1-pixels, fin.
4. Ir al paso 1.

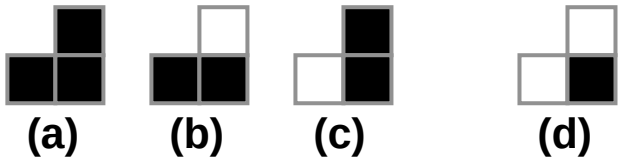




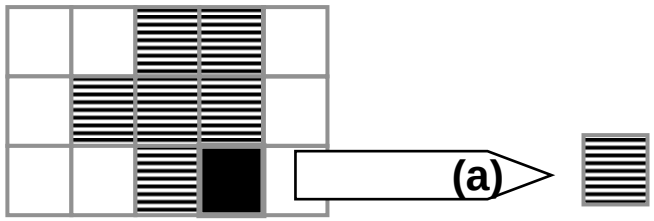
Algoritmo secuencial

- Para etiquetar un pixel sólo hace falta considerar sus vecinos anterior y superiores.
- Depende del tipo de conectividad que se utilice para objeto (aquí 4-conectividad).

mismo objeto | objeto nuevo



¿qúe ocurre en estos casos?



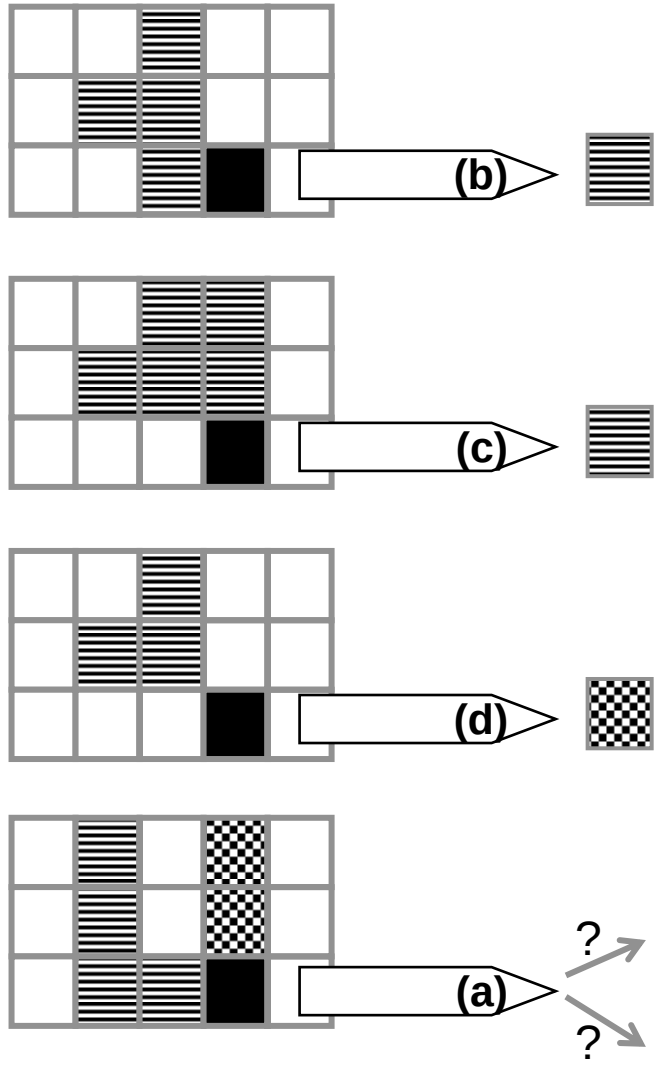


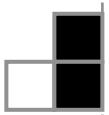
Tabla de Equivalencia

Alg. secuencial (4-conectividad)

- Procesar la imagen de izquierda a derecha, de arriba hacia abajo.

1. Si el siguiente pixel a procesar es 1-pixel:

ya procesados



1. Si sólo uno de sus vecinos (superior e izquierdo) es 1-pixel, copiar su etiqueta.

2. Si ámbos lo son, y tienen la misma etiqueta, copiarla.

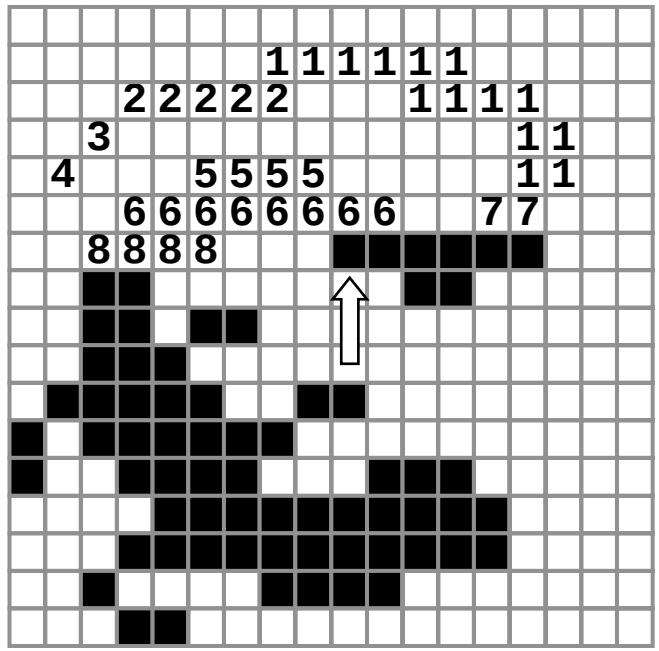
3. Si tienen etiquetas diferentes:

¿superior? ¿menor?

1. Copiar la del anterior.
2. Reflejar en la tabla la equivalencia.

4. Dlc, asignar una etiqueta nueva.

2. Si hay más pixels, ir al paso 1.

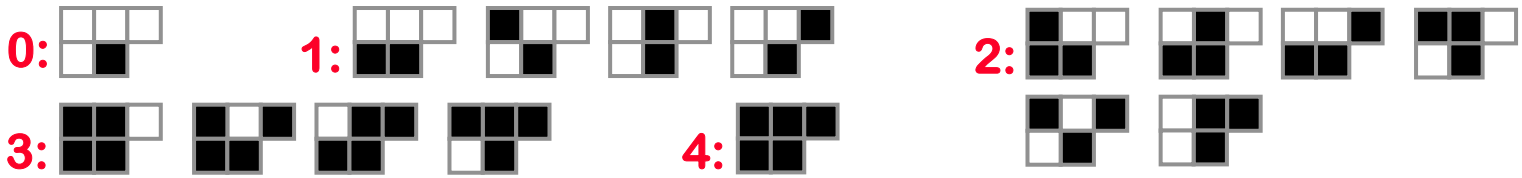



```
{1, 2, 7}
{3}
{4}
{5, 6, 8}
{}
```

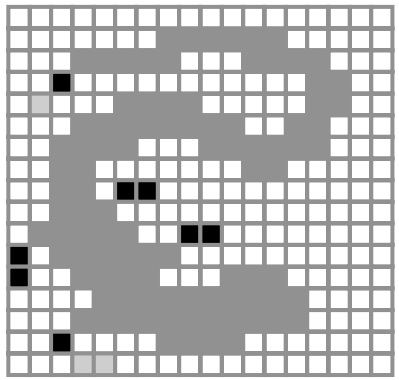
- Luego se reetiqueta con la menor de las etiquetas equivalentes.
- Los pixels de un mismo segmento siempre tienen la misma etiqueta

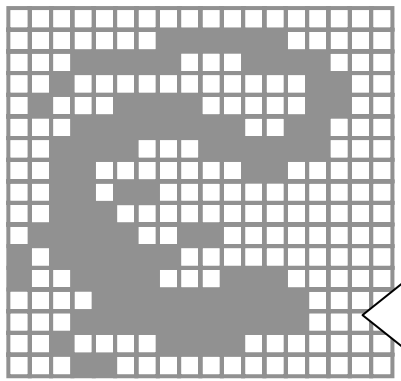
Algoritmo secuencial

- **8-conectividad:**



¿aparecerán vecinos con tres etiquetas diferentes?





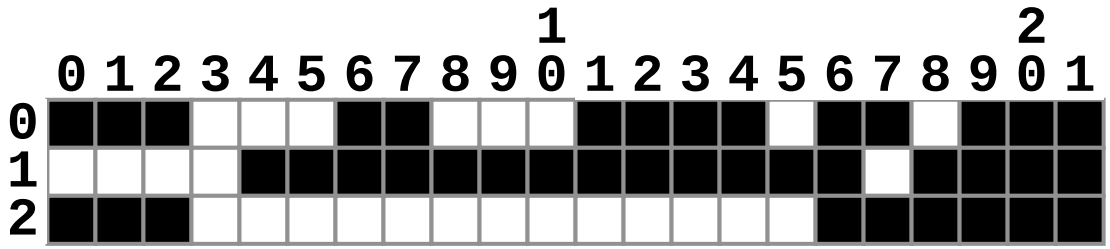
Conclusiones:

- Es conceptualmente sencillo y claro, pero la representación resultante es del mismo nivel que la imagen binaria
- No se ha extraído información adicional (descriptores, relaciones de inclusión, ...). La mayoría de los descriptores pueden calcularse simultáneamente.

Objetos elongados

Run Length Encoding (RLE)

- **RLE:** aprovecha la coherencia espacial de las imágenes binarias.



  - Varias posibilidades:

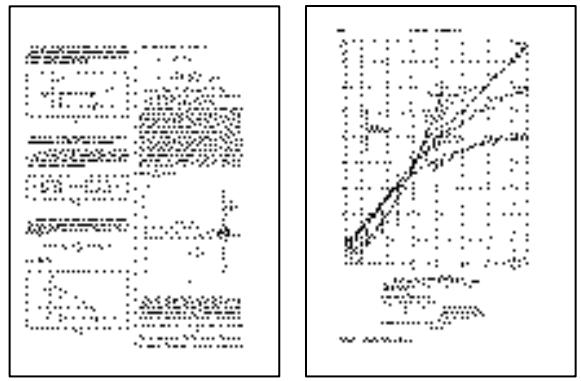
    - **A:** codificar la posición del primer 1-pixel y la cantidad de 1-pixels consecutivos:

```
(0,3) (6,2) (11,4) (16,2) (19,3)
(4,13) (18,4)
(0,3) (16,6)
```

    - **B:** codificar la longitud de cada segmento, comenzando por los 1-segmentos

```
3,3,2,3,4,1,2,1,3
0,4,13,1,4
3,13,6
```

  - Utilizado en transmisión de datos:

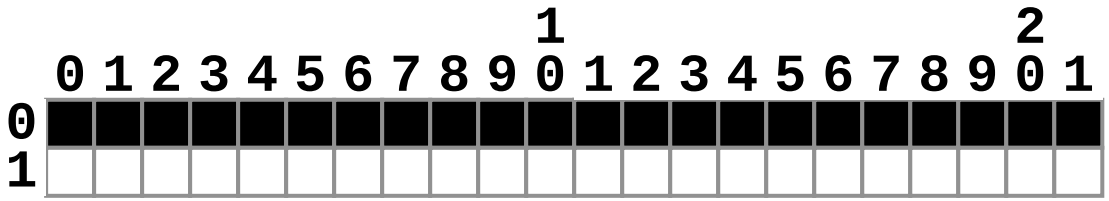


RLE

Ratio de compresión: tamaño original / tamaño comprimido

- **Mejor caso:**

A:	B:
(0,22) -	22 0

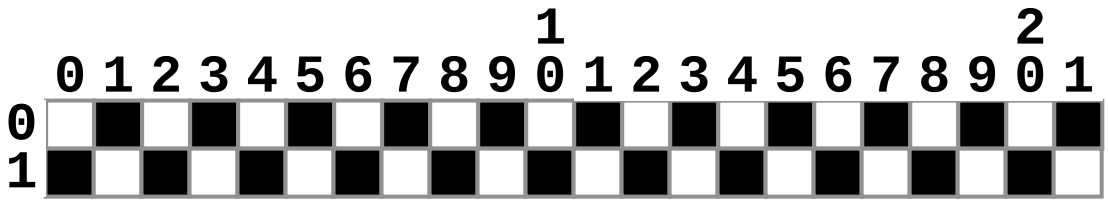


Ratio A: C=640 bits / 16 bits = 40

- **Peor caso:**

A: (1,1) (3,1)...

B: 0 1 1 1 1 ...



Ratio A: C bits / 16C bits = 0.0625

640 * 480 / 6874 * 16 = 2.793

Análisis de conectividad

1. El análisis puede hacerse por segmentos en vez de por pixels.

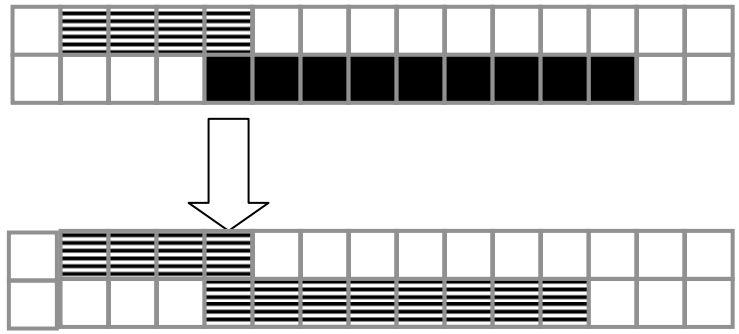
   Es mucho más eficiente.

2. Solo hace falta analizar dos filas de la imagen a la vez

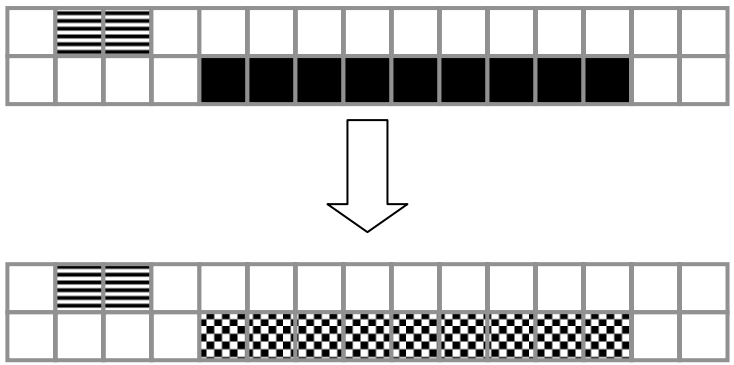
   Si en una fila no se actualiza un blob, no se volverá a actualizar

3. Los casos de conectividad se reducen a tres:

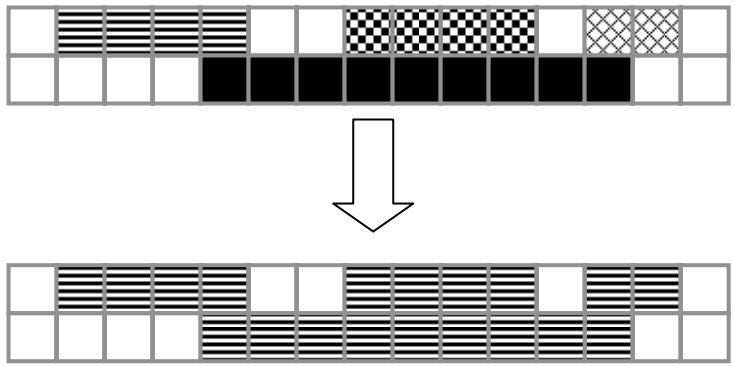
2. el mismo objeto



1. nuevo objeto

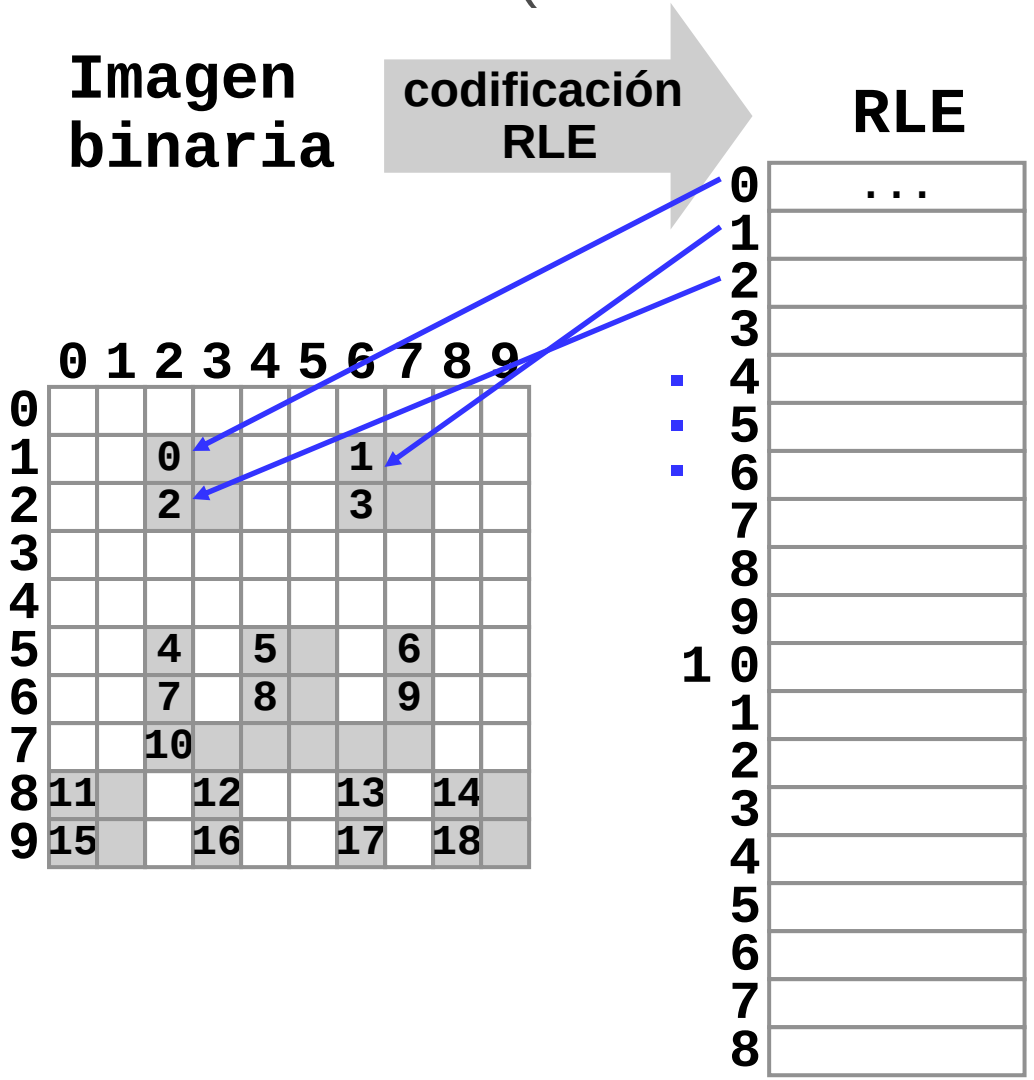


3. fusión de objetos



Esquema general

1. Obtener el RLE (sólo consideraremos 1-pixels):



Del RLE:

NFILAS
NSEGMENTOS

De una fila **f** del rle:

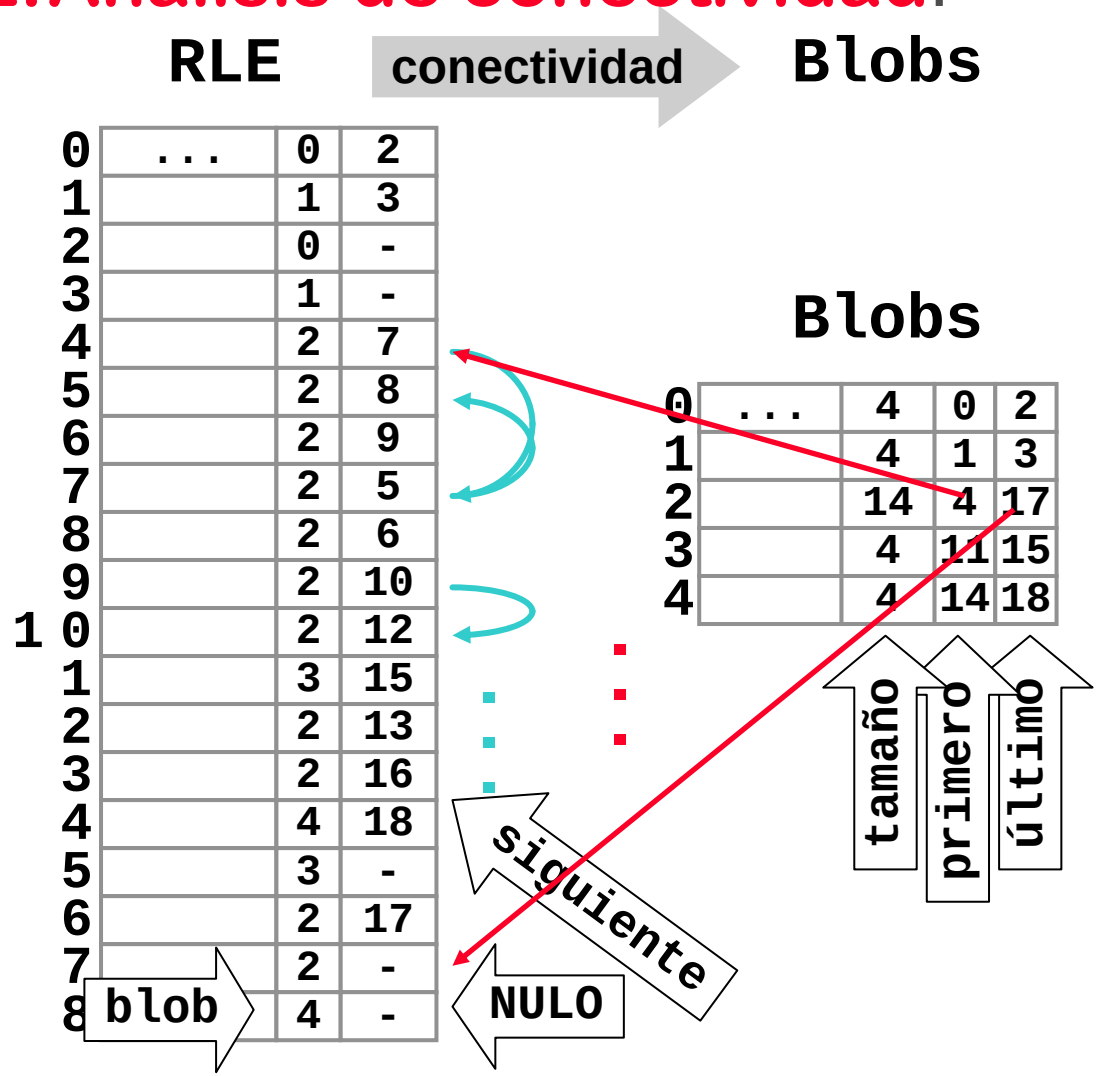
PRIMER_SEGMENTO(f)
ULTIMO_SEGMENTO(f)

De un segmento **s**:

FILA(s)
COLUMNA(s)
LONGITUD(s)
COMIENZO(s)
FINAL(s)

Esquema general

2. Análisis de conectividad:



De un segmento **s**:

BLOB(s)
SIGUIENTE(s)

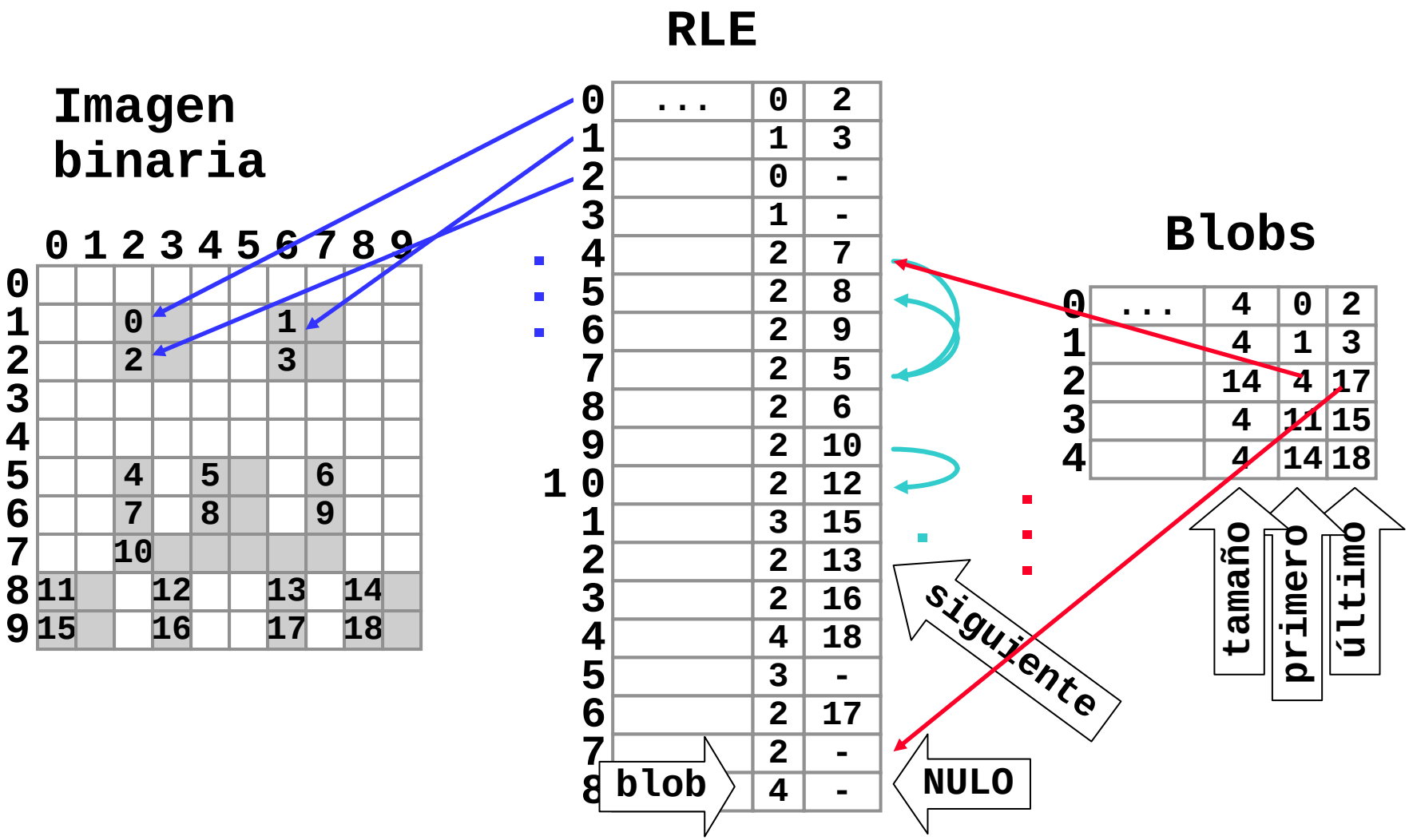
De un blob **b**:

TAMANO(b)
PRIMERO(b)
ULTIMO(b)

Para todos:

NULO (-1)

Esquema general



Algoritmo de conectividad

```
PARA cada fila f de la imagen HACER

      PARA cada segmento s de la fila f HACER

            [n, b] = blobs_que_toca(s);
            SI n = 0
               crear_blob(s)
            SINO SI n = 1
               anadir_segmento_a_blob(b0, s)
            SINO
               PARA cada blob bi de b excepto b0 HACER
                     fusionar_blobs(b0 ,bi)
               FPARA
               anadir_segmento_a_blob(b0, s)
            FSI
            FSI

      FPARA

FPARA
```

conectividad

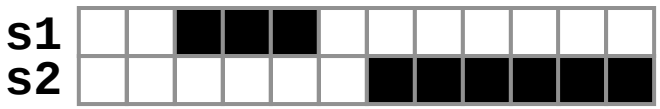
- 4-conectividad:

A_LA_IZQUIERDA(s1,s2)

SE_SOLAPAN(s1,s2)

A_LA_IZQUIERDA(s1,s2): (FINAL(s1) < COMIENZO(s2))

- 8-conectividad:



A_LA_IZQUIERDA(s1,s2)

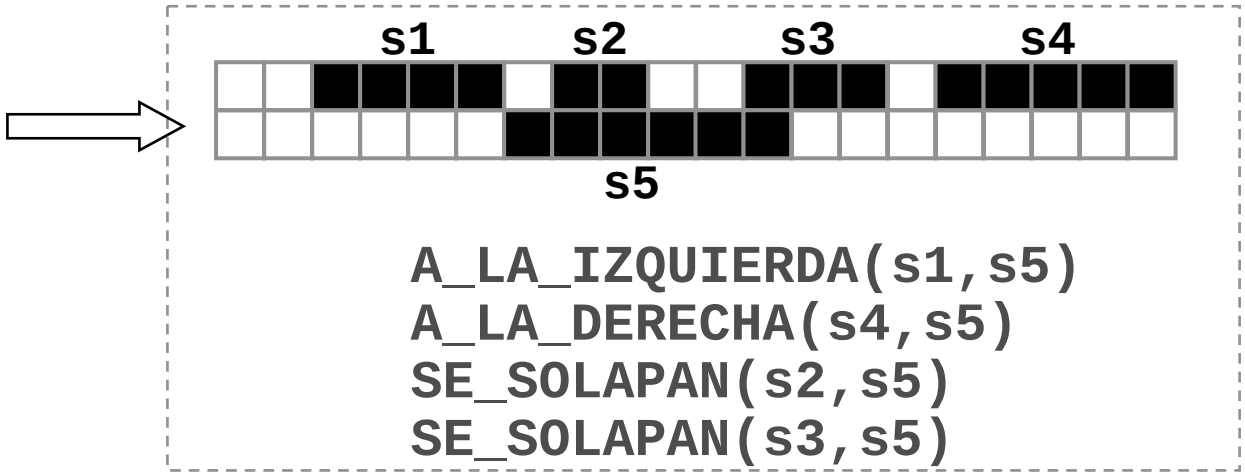
SE_SOLAPAN(s1,s2)

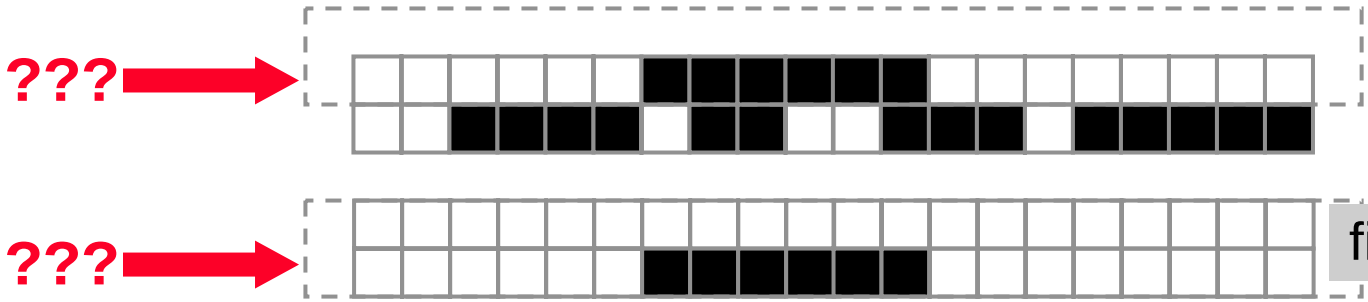
A_LA_IZQUIERDA(s1,s2): (FINAL(s1) + 1 < COMIENZO(s2))
(FINAL(s1) < COMIENZO(s2) - 1)

blobs_que_toca (4-conectividad)

- De dos segmentos **s1** y **s2**:

```
A_LA_IZQUIERDA(s1,s2):    (FINAL(s1) < COMIENZO(s2))
A_LA_DERECHA(s1,s2):      (A_LA_IZQUIERDA(s2,s1))
SE_SOLAPAN(s1,s2):        (!(A_LA_IZQUIERDA(s1,s2))
                       && !(A_LA_DERECHA(s1,s2)))
```





primera fila

fila anterior vacía

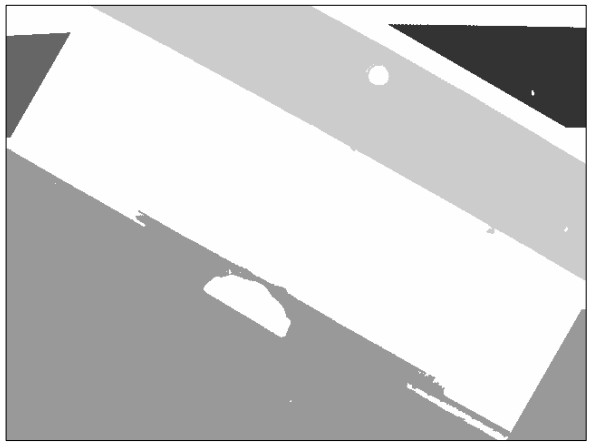
Filtro de tamaño

- Eliminación del ruido de tamaño y forma variable: si los objetos de interés tienen un tamaño de entre T_{min} y T_{max} pixels, el resto de los blobs pueden ignorarse.

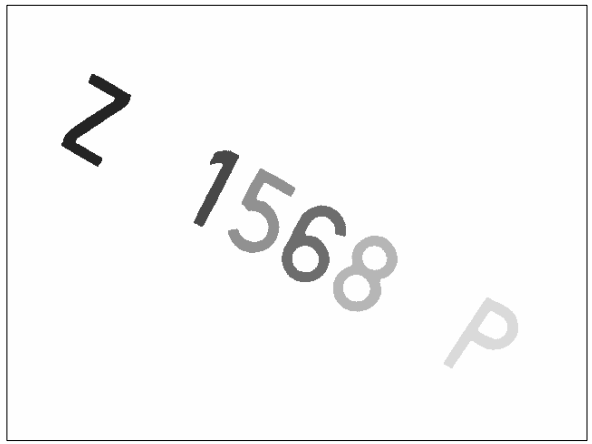
Imagen binaria

Menos de 400 pixels

Mas de 3000 pixels



Entre 400 y 3000 pixels



Filtro de tamaño

- Este análisis puede hacerse durante la conectividad:

```
crear_blob(s):

tamaño = LONGITUD(s);
```

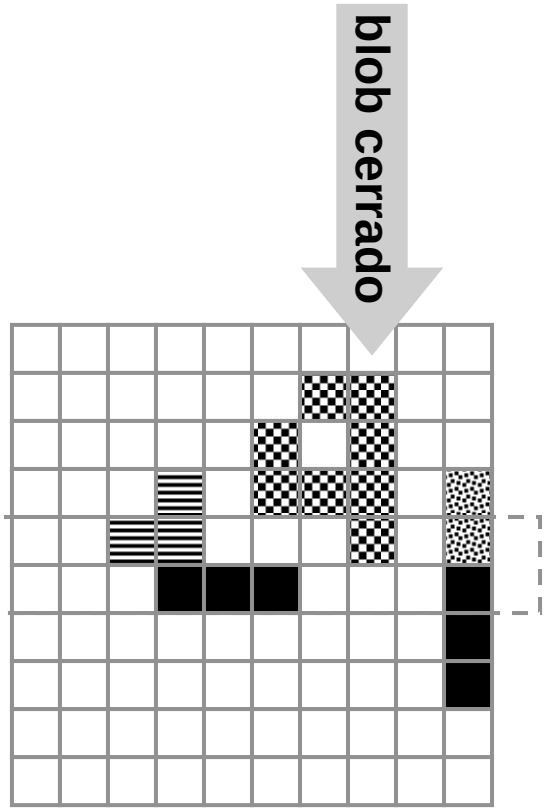
```
anadir_segmento_a_blob(b0, s):

tamano0 += LONGITUD(s);
```

```
fusionar_blobs(b0 , bi):

tamano0 += tamanoi;
```

- Un blob no actualizado en una fila, se puede considerar *cerrado*.

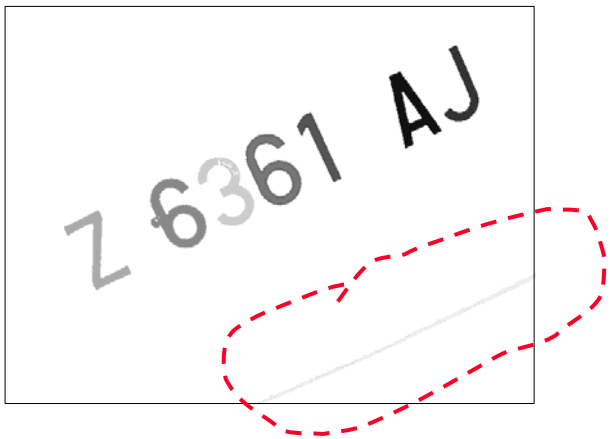


Filtro de tamaño

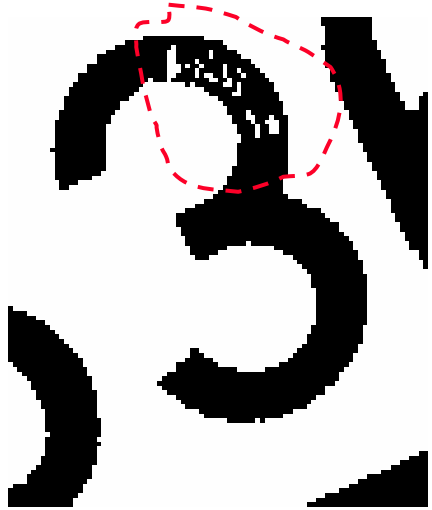
- No podemos resolver todos los problemas...

Imagen binaria

Entre 400 y 3000 pixels



- Ruido de sal:



- Solapamiento: