Lesson 2: Connectivity

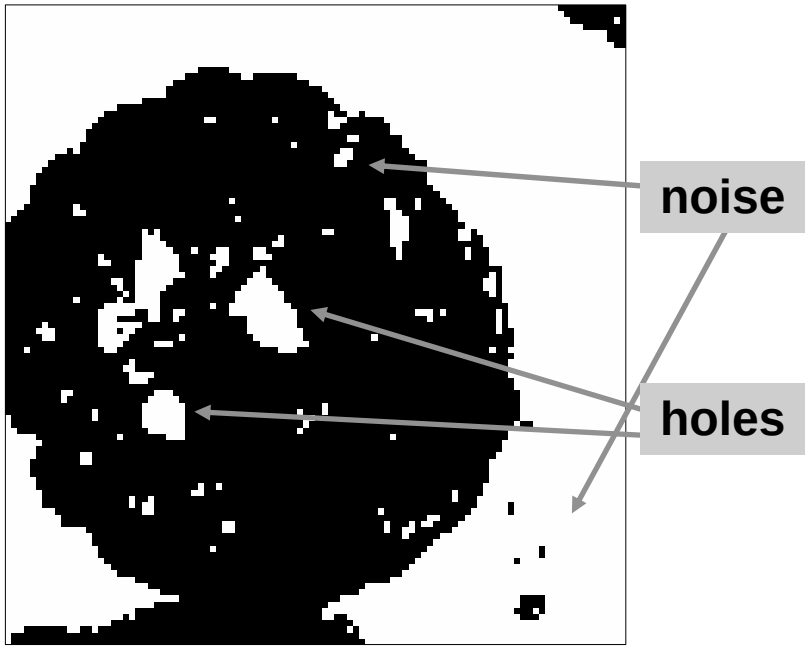
1. Definitions

2. Algorithms
   - Recursive
   - Sequential

3. Connectivity analysis
   - Run Length Encoding
   - Sequential algorithm

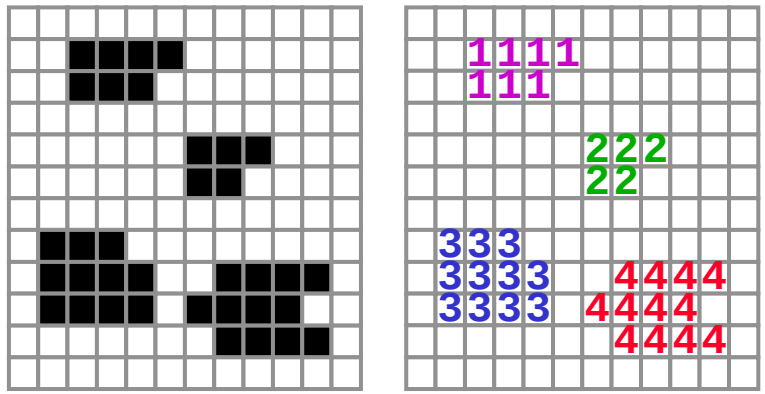
Connectivity

- **Purpose:** to separate objects in the scene:
  - From background
  - From each other
  - From holes



- There is salt and pepper noise, that we should try to eliminate

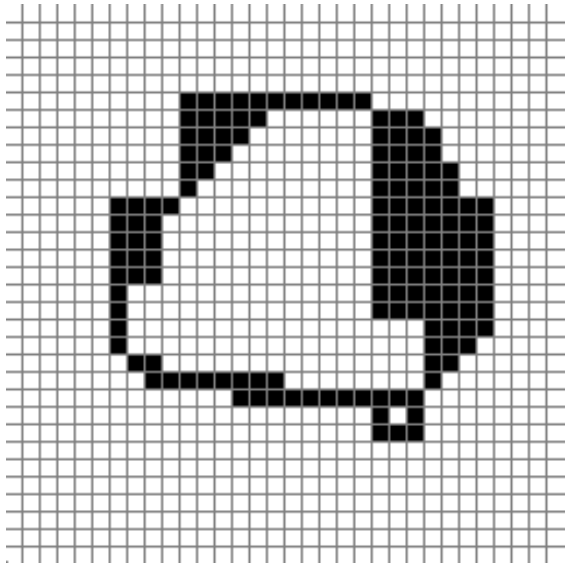
- Simple idea: label each region of contiguous pixels with a different value



- It uses the binary image
- It generates a colored image
  - ¿savings of space?
  - ¿semantic improvement?

Definitions: connectivity

- **Problem:**



how many objects?
holes, background?

- It depends on what you consider adjacent to a pixel.
- Also interior, exterior …

- **Connectivity (neighbors):** Spatial proximity between pixels in the binary image.

4-connectivity

	x_2	
x_3	x_0	x_1
	x_4	

8-connectivity

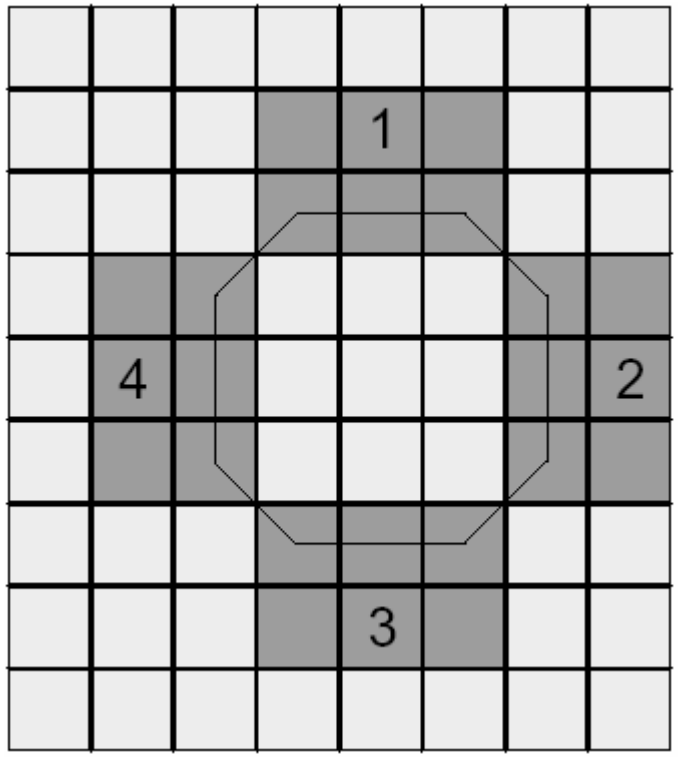
x_4	x_3	x_2
x_5	x_0	x_1
x_6	x_7	x_8

6-connectivity

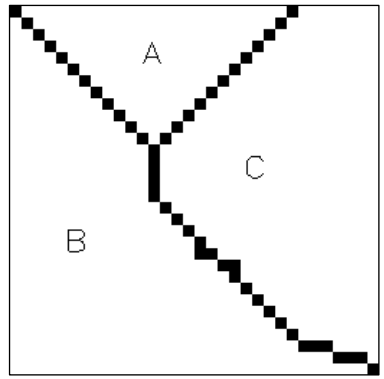
	x_3	x_2
x_4	x_0	x_1
x_5	x_6	

Connectivity

- **Euclidean geometry:** a closed curve divides the plane into two non-contiguous regions.
- Given a digital image, a simple closed curve (with no overlap) segments the image into two disjoint connected regions: interior and exterior.



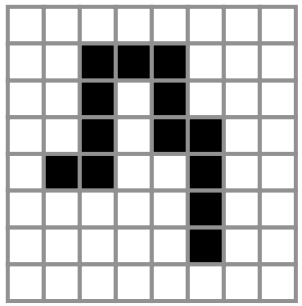
- **If we use 4-connectivity:**
  - Four objects
  - Two disjoint regions
  - But no Jordan curve!
- **If we use 8-connectivity:**
  - **One object**
  - **A Jordan curve inside the object**
  - **No two disjoint regions !!**



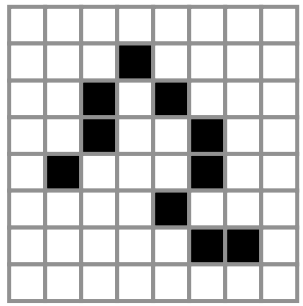
Definitions

- **Foreground:** set of 1-pixels (S)
- **Path:** a path from pixel $[i_0, j_0]$ to pixel $[i_n, j_n]$ is a sequence of pixels such that $[i_k, j_k]$ is a neighbor of $[i_{k+1}, j_{k+1}]$ for each $k=0...n-1$

4-connected



8-connected



- **Connectivity:** two pixels p, q $\in$ S are connected if there is a path from p to q whose pixels $\in$ S.
- **It's an equivalence relation:**
  1. **Reflexivity:** p is connected to p
  2. **Symmetry:** If p is connected to q, then q is connected to p
  3. **Transitivity:** If p is connected to q, and q is connected to r, then p is connected to r

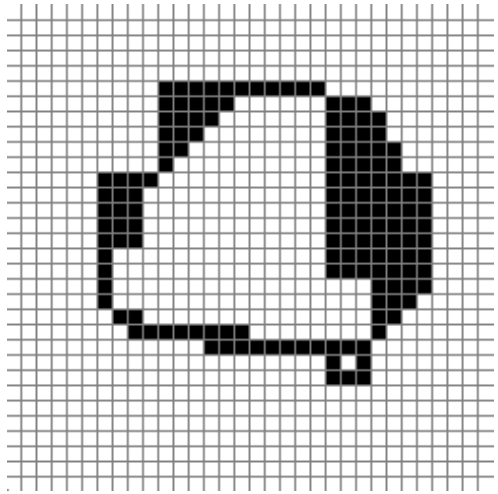
Definitions

- **Connected component (blob):** set of mutually connected pixels.

Each blob should correspond to a different object.

- **Background:** set of connected components of the complement of S ($\overline{S}$) such that they contain pixels in the image limits.
- **Holes:** set of connected components of $\overline{S}$ that DO NOT have pixels in the image limits.

- Returning to the original problem...

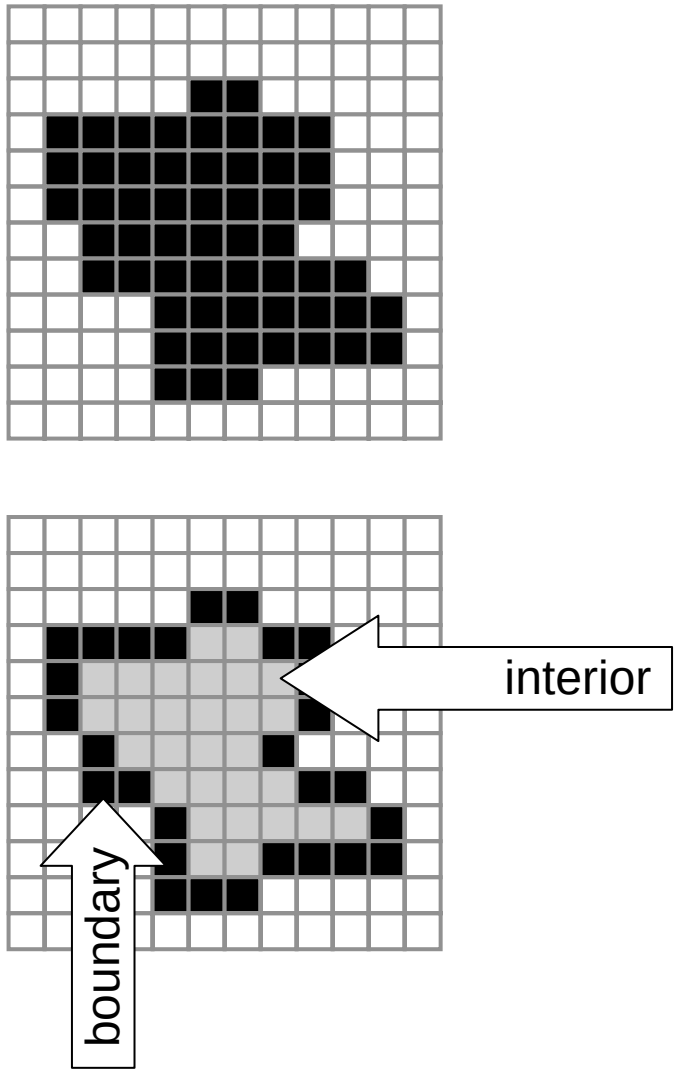


S	$\overline{S}$	Objs	Holes
4	4	4	2
8	8	1	1
6	4	?	?

In order to avoid inconsistencies, we use one connectivity for foreground and another for the background, or six-connectivity for both?

Definitions

- **Boundary:** the boundary of S is the set of pixels with neighbors in $\overline{S}$ (usually denoted as S').
- **Interior:** set of pixels from S that do not belong to S' (S - S').
- **To surround:** a region T surrounds S if any path from any pixel of S to the image limit intersects T.



Labeling algorithms: recursive

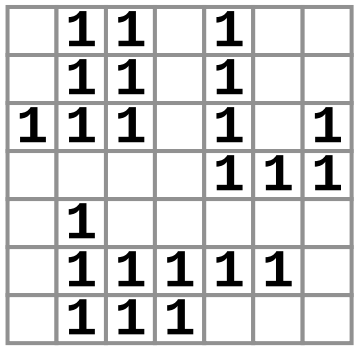
- **Labeling:** to partition the image into connected components.

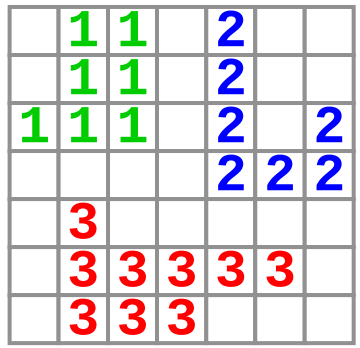
We hope that connected components correspond to different objects

- A different label is assigned to each set of pixels of each connected component.
  - **Recursive:** simple but inefficient in sequential machines (it's parallelizable).
  - **Sequential:** requires two passes, but only two rows of the image (used when there is limited storage).

Recursive algorithm:

1. Look for next 1-pixel and assign a new label L.
2. Recursively assign L to all its 1-pixel neighbors.
3. If no more 1-pixels, end.
4. Go to step 1.

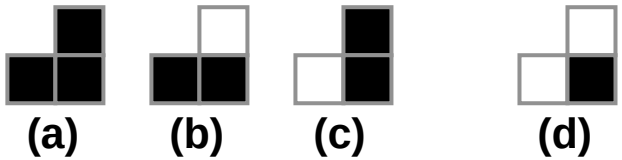




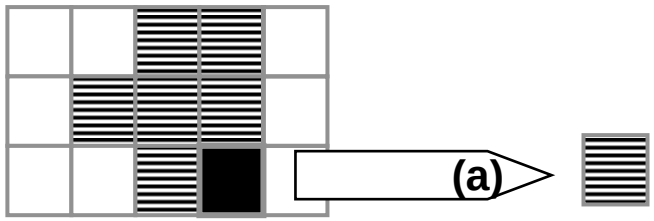
Sequential Algorithm

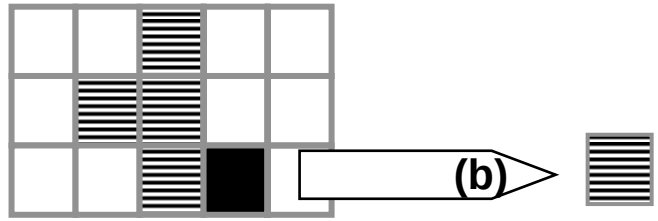
- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

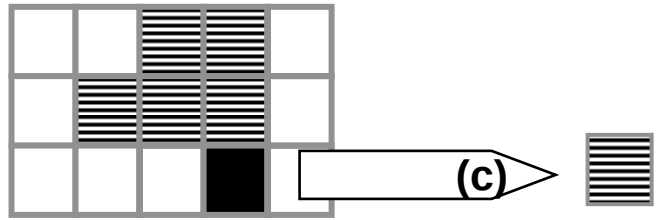
Same object | New object

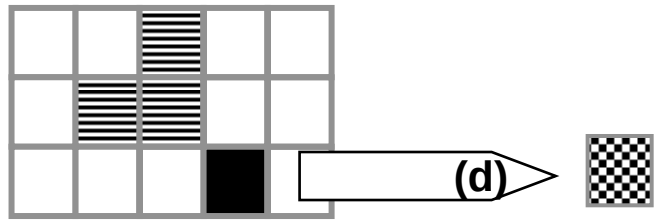


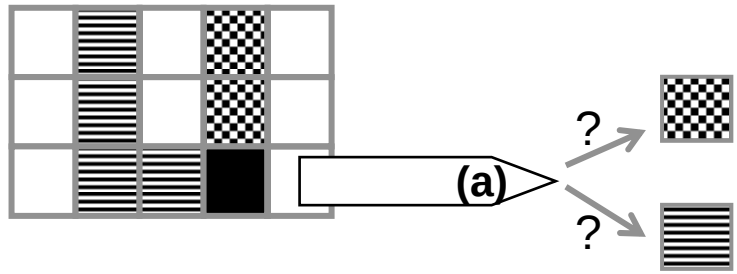
What happens in these cases?











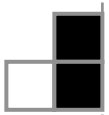
Equivalence table

Sequential alg. (4-connectivity)

- Process the image from left to right, top to bottom.

1. If the next pixel to process is 1-pixel:

Already processed



1. If only one of its neighbors (superior or left) is 1-pixel, copy its label.

2. If both are, and have the same label, copy it.

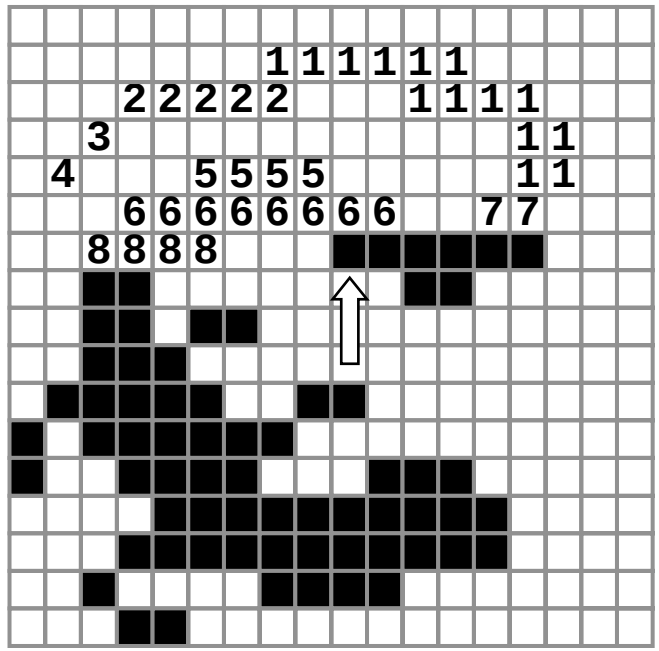
3. If they have different labels:

superior? smallest?

1. Copy the label from the prior.
2. Reflect the change in the table of equivalences.

4. Otw, assign a new label.

2. More pixels? Go to step 1.

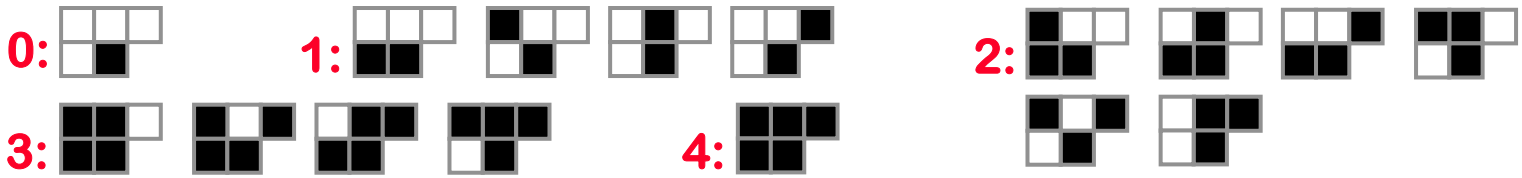



```
{1, 2, 7}
{3}
{4}
{5, 6, 8}
{}
```

- Re-label with the smallest of equivalent labels.
- Pixels of the same segment always have the same label.

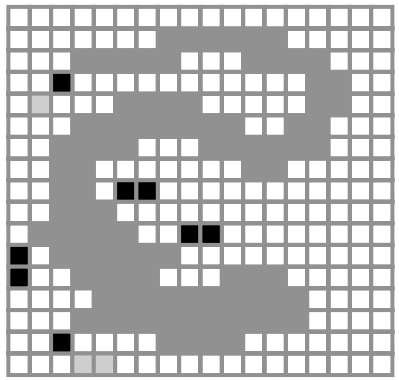
Sequential Algorithm

- **8-connectivity:**

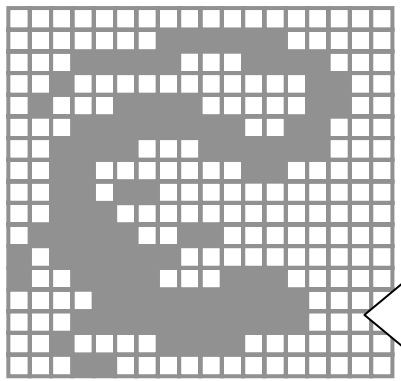


Will neighbors with three different labels show up?

4-connect.



8-connect.



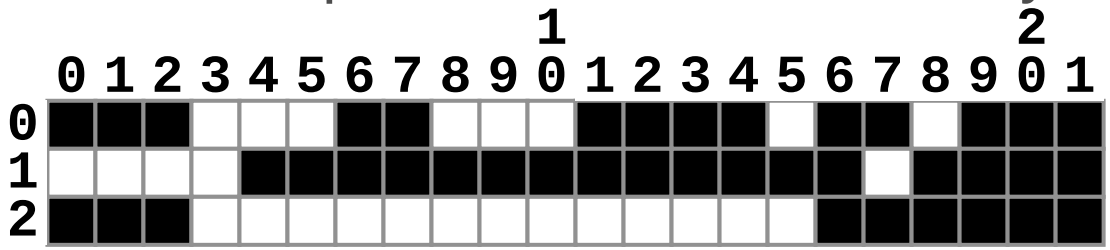
Conclusions:

- Conceptually simple and clear, but the resulting representation is of the same conceptual level to that of the binary image.
- No additional information has been obtained (descriptors, topological relations, ...). Most descriptors can be computed at the same time.

Elongated objects

Run Length Encoding (RLE)

- **RLE:** makes use of spatial coherence in binary images.



  - Several alternatives:

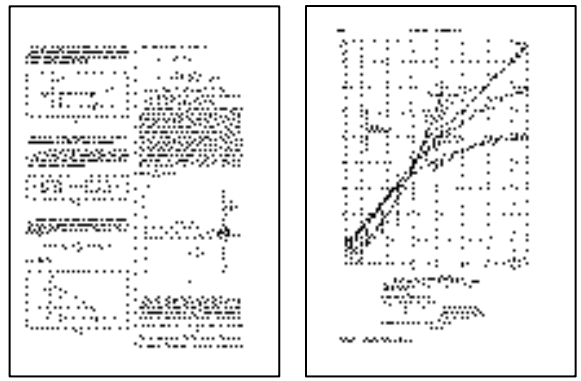
    - **A:** codify the position of the first 1-pixel as well as the amount of consecutive 1-pixels:

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

    - **B:** codify the length of each segment, beginning with 1-segmentos

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

  - Used in data transmission:



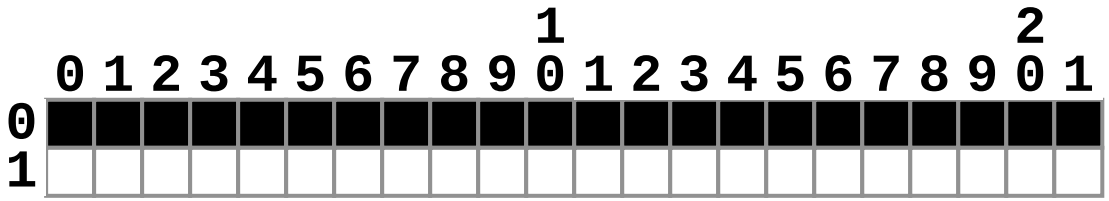
RLE

Compression ratio: original size / compressed size

- **Best case:**

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

```
B:
22
0
```

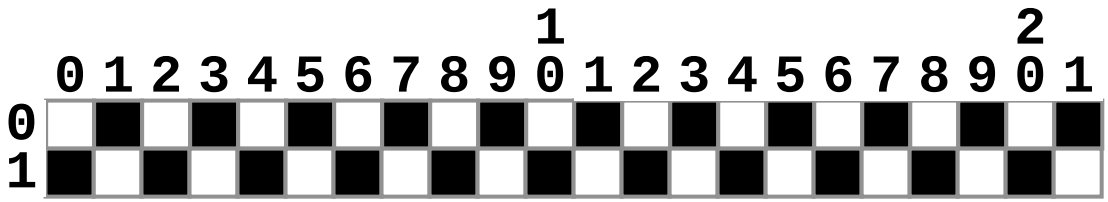


Ratio A: C=640 bits / 16 bits = 40

- **Worst case:**

```
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

Connectivity Analysis

1. Can be carried out segment-wise instead of pixel-wise.

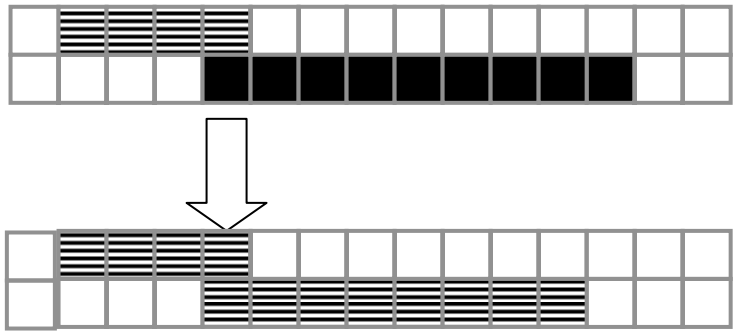
   Much more efficient.

2. Only two rows of the image are required at a certain step

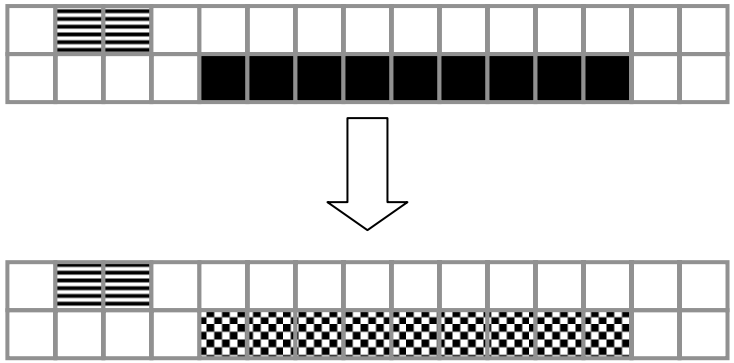
   If a blob is not updated for a whole row, it will never more be updated.

3. Connectivity cases are reduced to three:

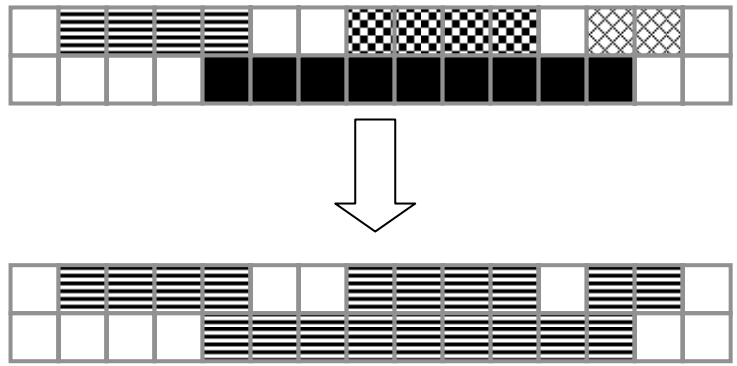
2. Same object



1. New object

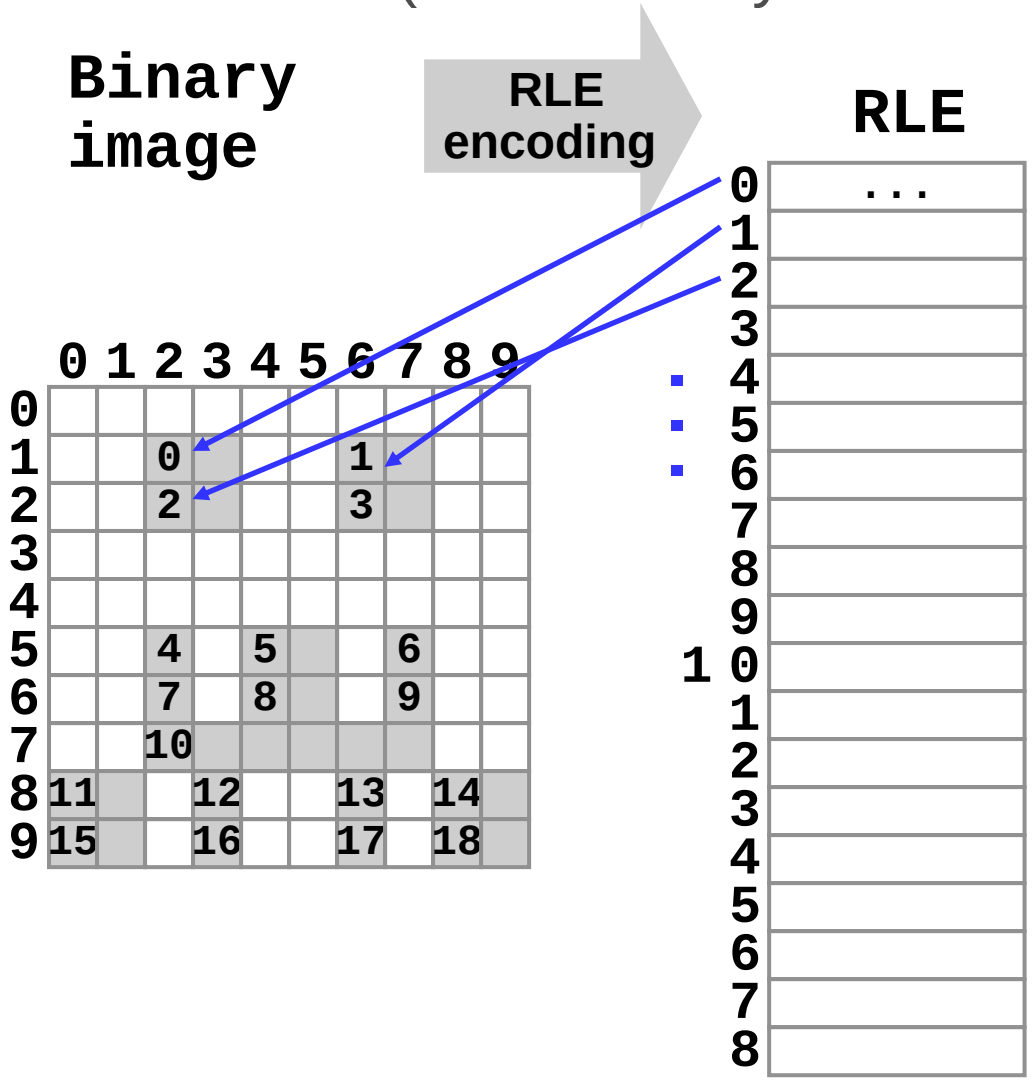


3. Object fusion



General scheme

1. Obtain RLE (we shall only consider 1-pixels):



Related to RLE:

NFILAS
NSEGMENTOS

Related to row **f** of rle:

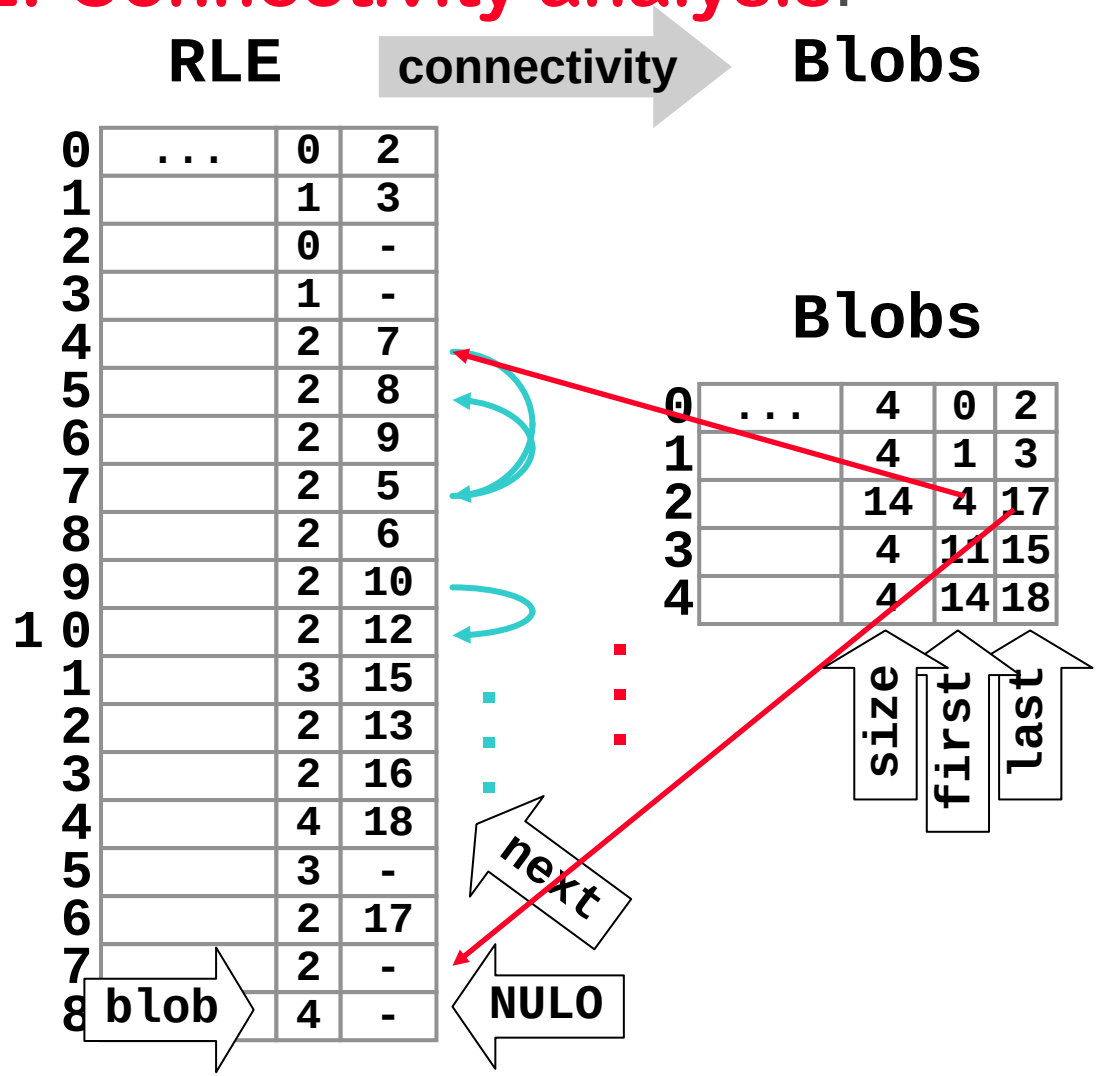
PRIMER_SEGMENTO(f)
ULTIMO_SEGMENTO(f)

Related to segment **s**:

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

General scheme

2. Connectivity analysis:



Related to segment **s**:

BLOB(s)
SIGUIENTE(s)

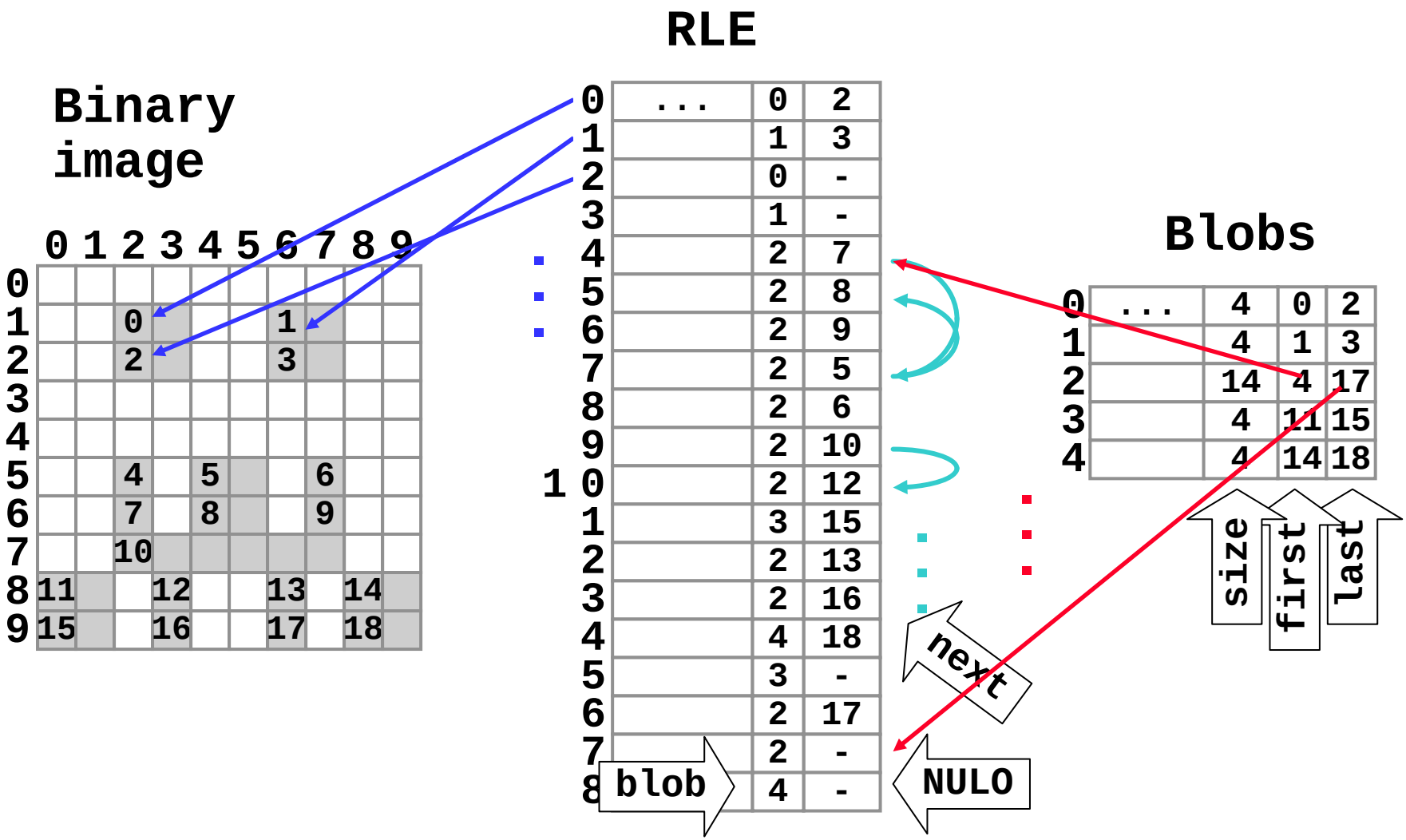
Related to blob **b**:

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

For all:

NULO (-1)

General scheme



Connectivity algorithm

```
FOR each row f in the image DO

      FOR each segment s of row f DO

            [n, b] = blobs_que_toca(s);
            IF n = 0
               crear_blob(s)
            ELSEIF n = 1
               anadir_segmento_a_blob(b0, s)
            ELSE
               FOR each blob bi of b except b0 DO
                     fusionar_blobs(b0 ,bi)
               ENDFOR
               anadir_segmento_a_blob(b0, s)
            ENDIF

    ENDFOR

ENDFOR
```

connectivity

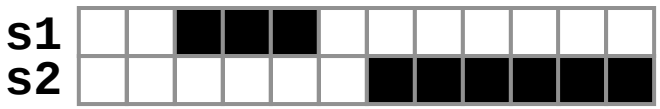
- 4-connectivity:

A_LA_IZQUIERDA(s1,s2)

SE_SOLAPAN(s1,s2)

A_LA_IZQUIERDA(s1,s2):

- 8-connectivity:



A_LA_IZQUIERDA(s1,s2)

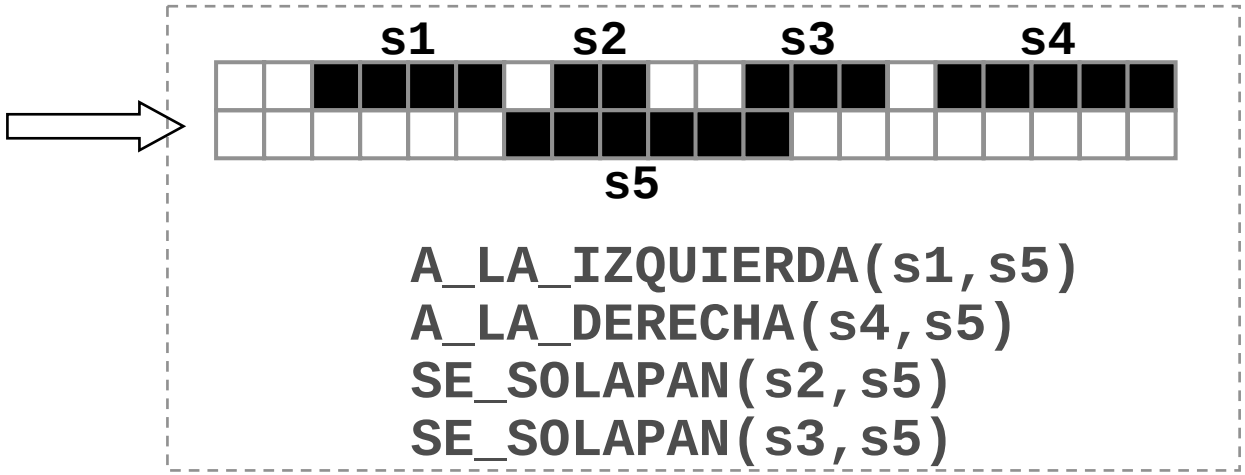
SE_SOLAPAN(s1,s2)

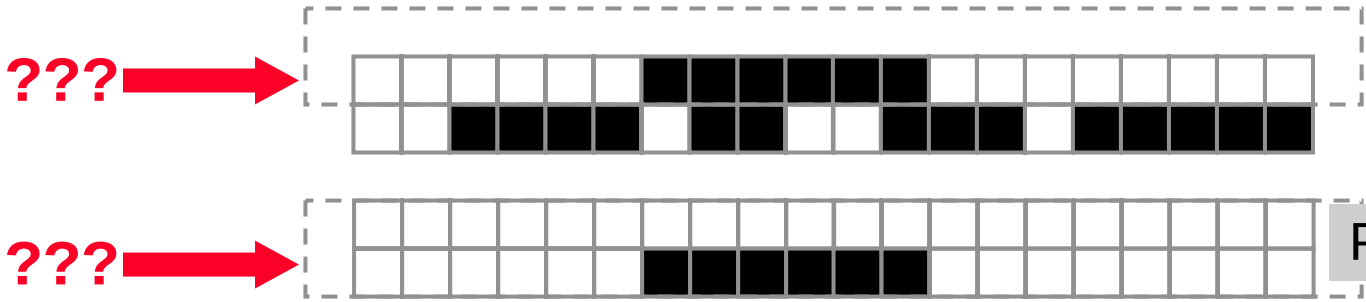
A_LA_IZQUIERDA(s1,s2):

blobs_que_toca (4-connectivity)

- Given two segments **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)))
```





First row

Prior row empty

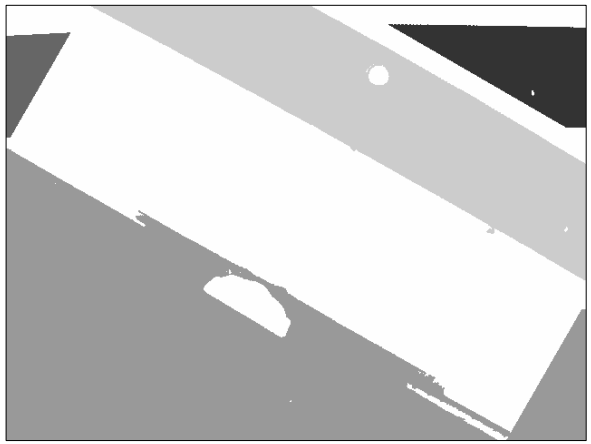
Size filter

- Elimination of noise of variable size and shape: if the objects of interest have a size between T_{min} and T_{max} pixels, all other blobs can be ignored.

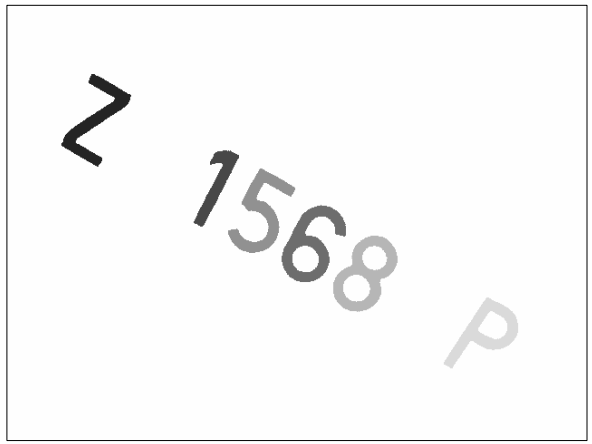
Binary image

Less than 400 pixels

More than 3000 pixels



Between 400 and 3000 pixels



Size filter

- This analysis can be carried out during connectivity:
- A blob not updated while processing a row can be considered *closed*.

```
crear_blob(s):

tamaño = LONGITUD(s);
```

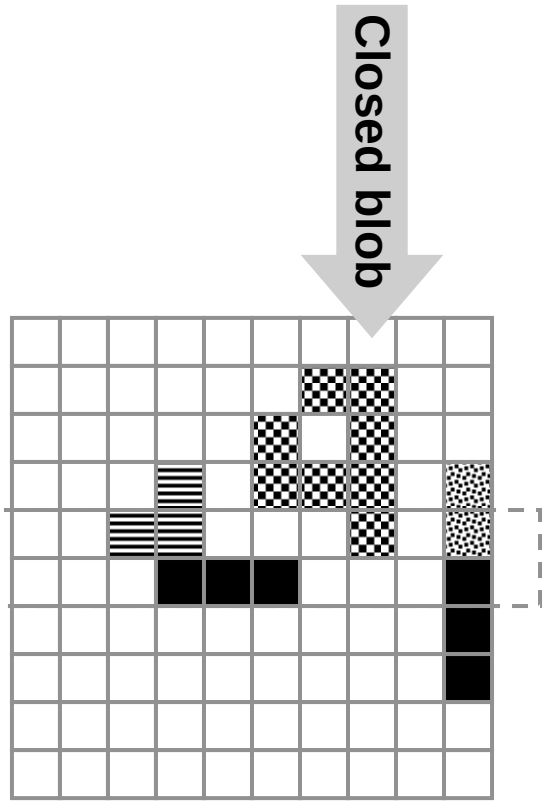
```
anadir_segmento_a_blob(b0, s):

tamano0 += LONGITUD(s);
```

```
fusionar_blobs(b0 ,bi):

tamano0 += tamanoi;
```

processing

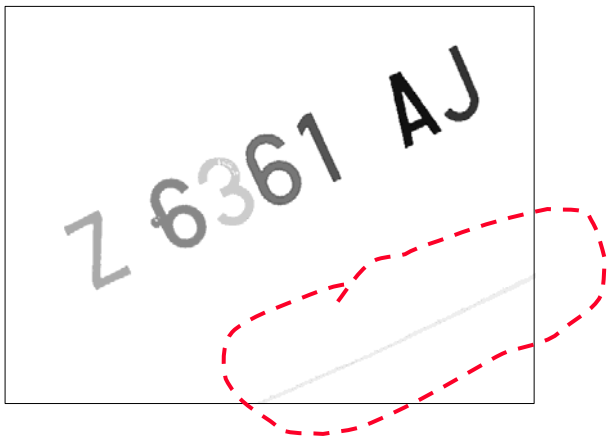


Size filter

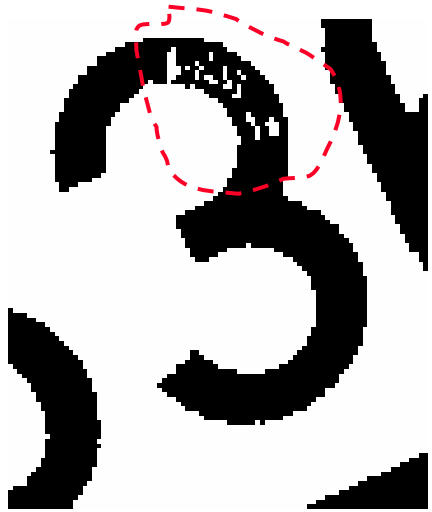
- Not all problems can be solved...

Binary image

Between 400 and 3000 pixels



- Salt noise:



- Overlap: