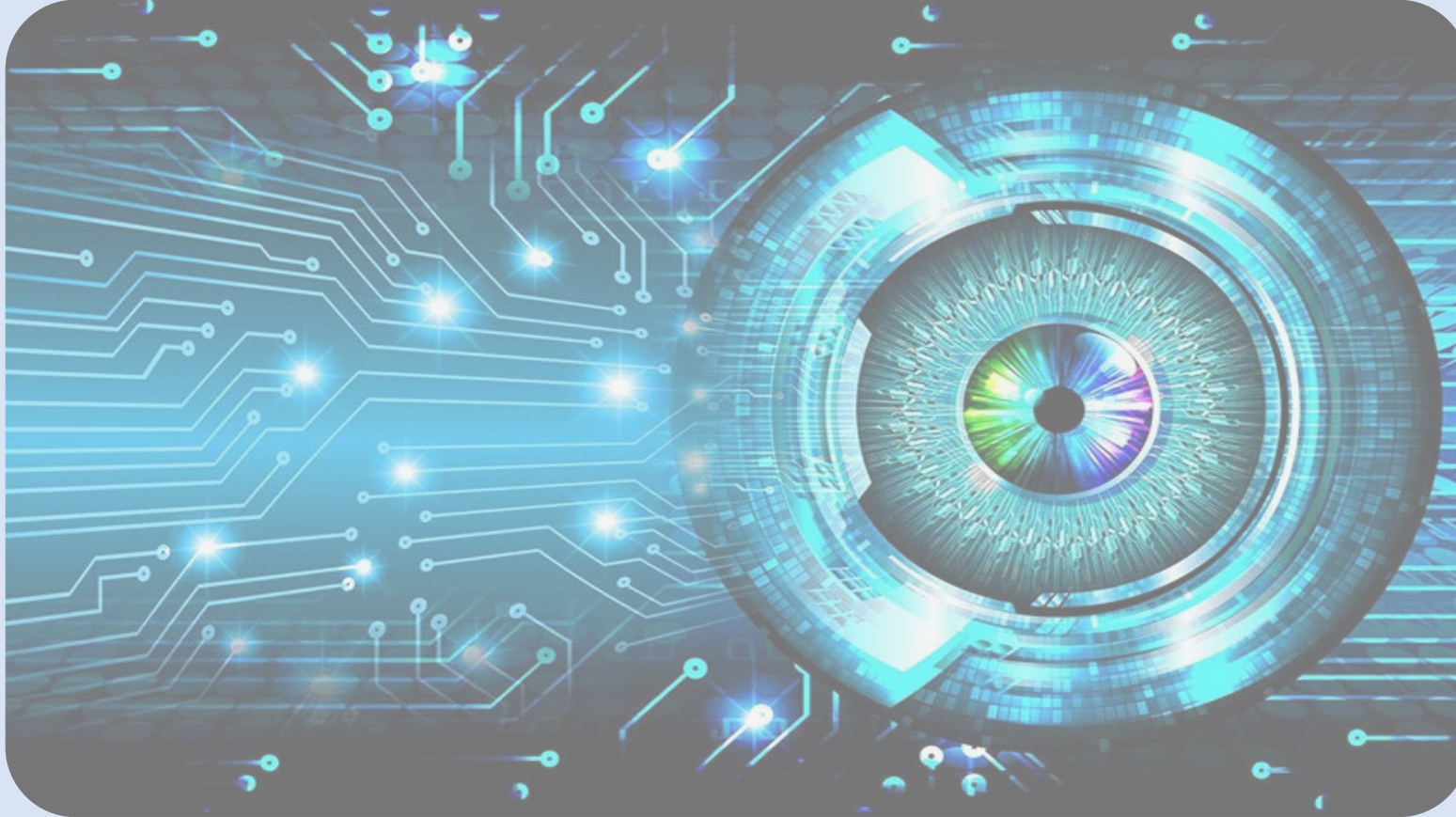


Computer Vision CSC 3014

Lecture 08 and 09



Dr Danish Mahmood Khan
Senior Lecturer
danishmk@imail.sunway.edu.my

Course Plan

Week	Lecture Topic	Practical	Remark
1	• Introduction to Computer Vision	No Lab	
2	• Image Capture and Storage	No Lab	Literature Review & Project Briefing will be in Class)
3	• Advanced Morphological Operations	Project Selection & Start of Literature Review	Wesak Day Holiday (Briefing will be in Class)
4	• Image Segmentation - Edge-based	Lab 1	
5	• Image Segmentation - Thresholding	Lab 2	
6	• Image Segmentation - Region-based	Literature Review Finalization	Agong's Birthday Holiday
7	• Image Segmentation - Watershed	Lab 3	Literature Review Submission (13 th June 2025) Start Project Implementation
	MID TERM BREAK		
8	• Shape Representation and Description	Lab 4	
9	• Class Test (02/07/2025)	Lab 5	
10	• Shape Representation and Description	Lab 6	
11	• Pattern Recognition	Lab 7 & 8	
12	• Pattern Recognition	Lab 9	
13	Project Evaluation (28 th July 2025)		

Final Assessment:	Examination (50%)
Coursework:	Class Test (10%) Project: Literature Review (20%) Project: Development (20%)
Total	100%
Coursework Details:	Literature Review Submission – Week 7 (13/06/2025) Class Test – Week 9 (02/07/2025) 9 th July 2025 Group Project Submission – Week 12 (25/07/2025) Group Project Evaluation – Week 13 (28/07/2025)

Assessment Dates / Marks Allocation

Assessment Methods	Marks Breakdown	Important Date	Component
Literature Review	20%	Project Selection & Start of Literature Review: Week 03 Literature Review Submission: Week 07 – 13th June 2025	Course Work
Class Test	10%	Week 09 – 2nd July 2025 (During Lecture) Week 10 – 9th July 2025 (During Lecture) MCQs Covers until Week 7	
Project	20%	Start of Project Implementation: Week 07 Group Project Submission – Week 12 (25th July 2025) Group Project Evaluation (During Lab Session) – Week 13 (28th July 2025)	
Written Exam	50%	As per Examination Timetable	Final Exam



REMINDER

✓ Coursework Compulsory Pass

- Students must pass both the coursework and final project components in order to pass the course.
- The passing grade is 40/100

✓ Lab Sessions

- Lab sessions begin in Week 3.

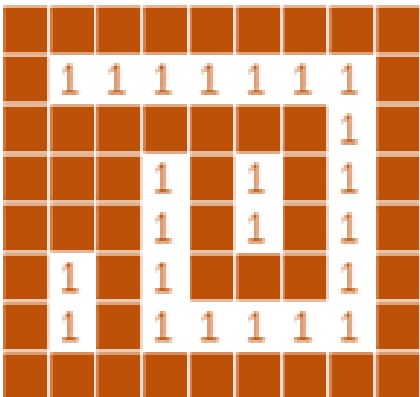
Region Labelling

Region labelling (or connected component labelling) is a fundamental image processing technique used to **identify**, and **label connected regions** (components) in **binary** or **grayscale** images. It is typically applied to **binary images** but can also be extended to **grayscale images** using techniques like **region growing**, **threshold-based** segmentation etc.

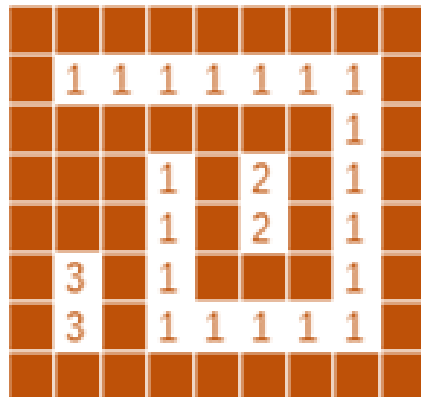
EXAMPLES

- ❖ Object counting (e.g., cells in microscopy)
- ❖ OCR (Optical Character Recognition)
- ❖ Medical image analysis

Thresholded image:



Labelled image:



PRE-
CONDITION

The image must be:

- ❖ Binary (e.g., 0 = background, 1 = foreground)
- ❖ Foreground objects are typically white (1), background black (0)

CONNECTIVITY

- ❖ 4-connectivity: Neighboring pixels in vertical and horizontal directions
- ❖ 8-connectivity: Also includes diagonal neighbors

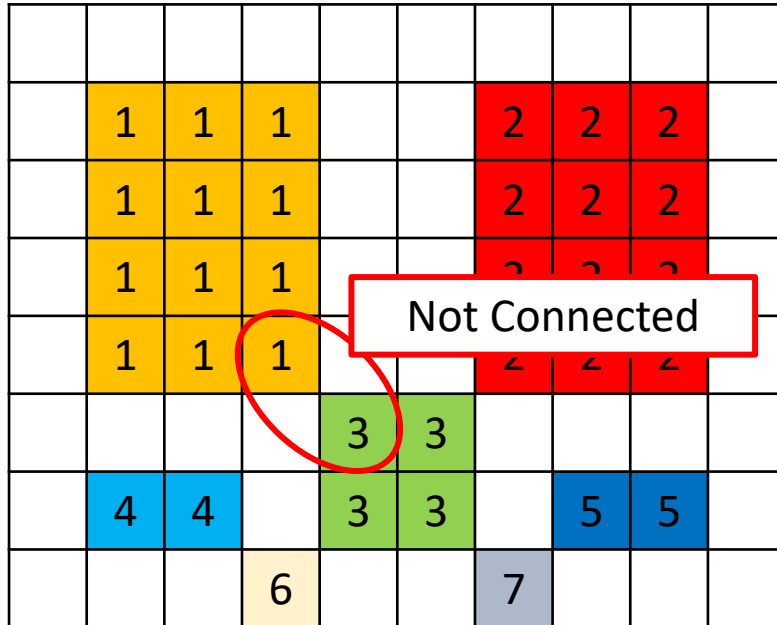
Note: The choice of connectivity affects whether diagonally touching pixels are considered part of the same region.

Region Labelling

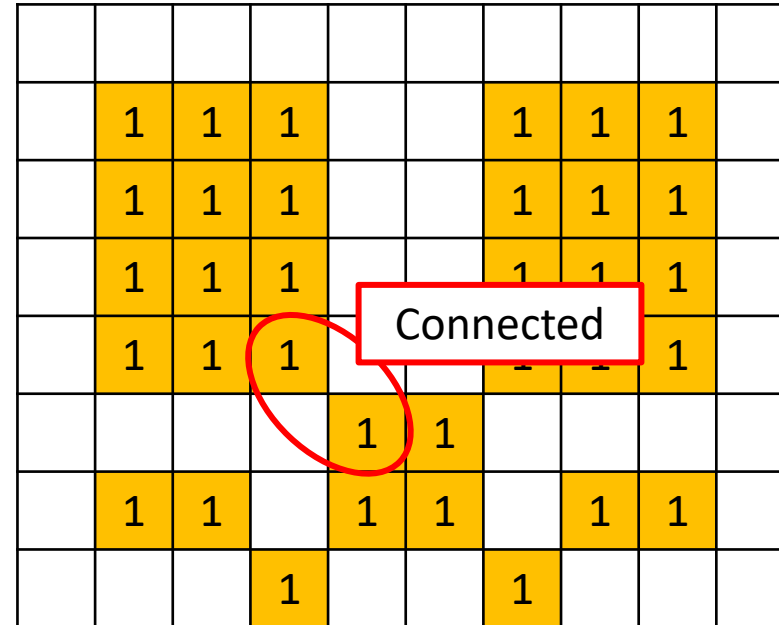
- ❖ You could skip this step if you used region growing or split-and-merge to perform segmentation because these methods label the regions during the process. You can use this labeling to identify the number and location of the regions.
- ❖ However, this step can still be useful in other scenarios, such as when you are trying to identify and label markers for watershed segmentation, or when you have segmented all the regions using thresholding (regions extracted using thresholding are not labeled by default).

N4 or N8 Region Identification

- You can choose whether a pixel should have only 4 neighbours (4-connectivity) or 8 neighbours (8-connectivity).
- Diagonally connected pixels will not be grouped in cases we assume a pixel should have only 4 neighbours (4-connectivity).



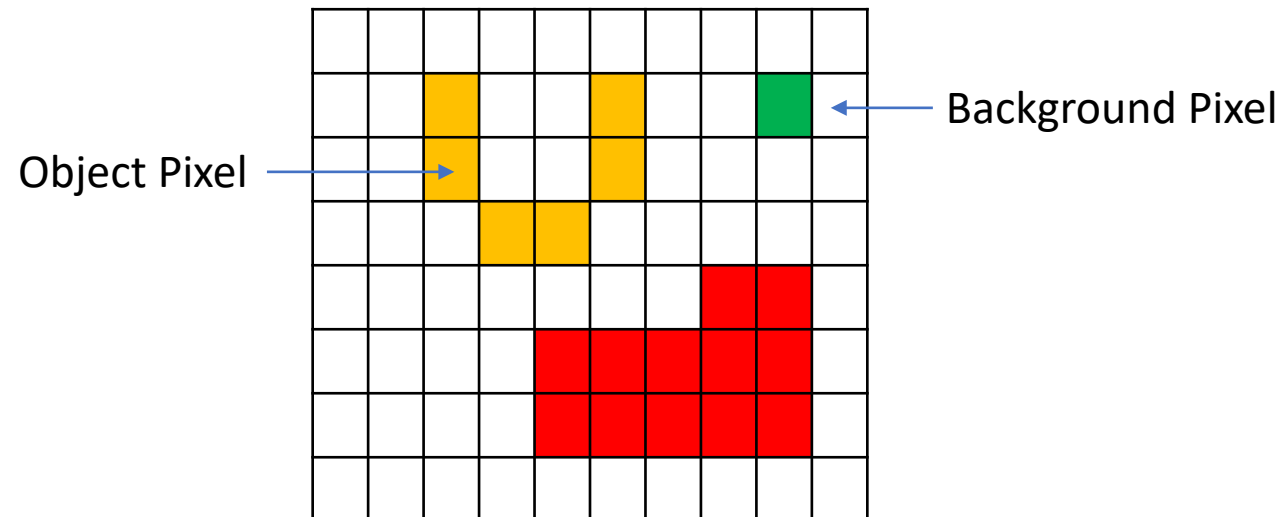
4-Connectivity (N4)



8-Connectivity (N8)

Region Labelling

- A two-pass algorithm involves checking the neighbors of every non-zero pixel in the binary image.
- We will call the non-zero pixel (i.e., with a value of '1') the **object pixel**, and the zero pixel (i.e., with a value of '0') the **background pixel**.
- The number of neighbors that need to be evaluated depends on whether you choose to work with 4-connectivity or 8-connectivity.

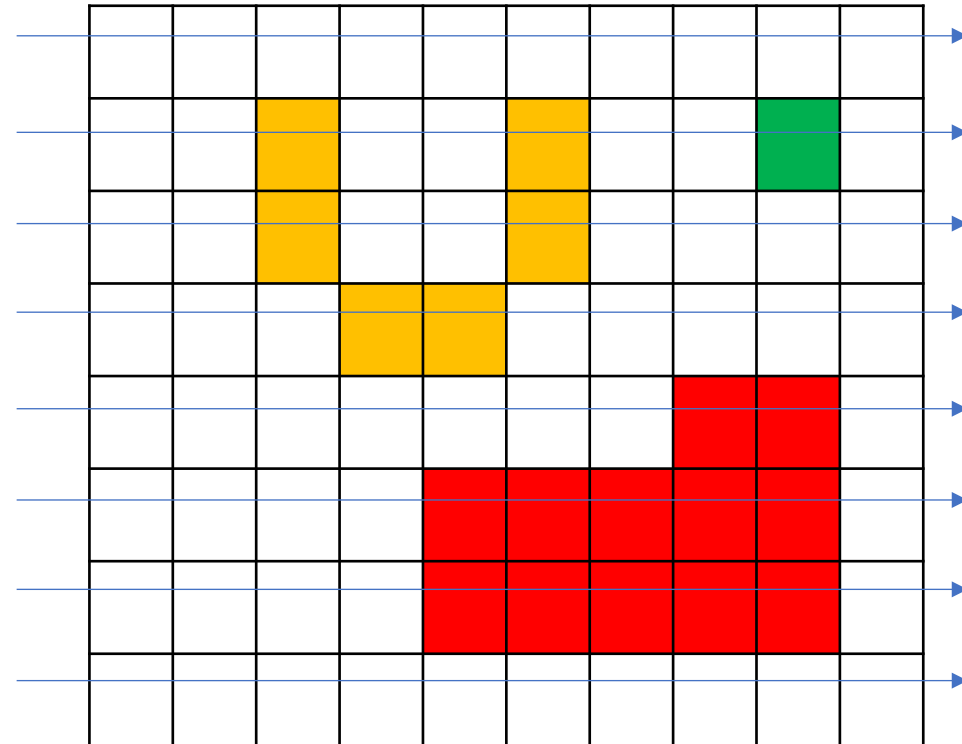


Region Labelling

- In this example, assuming that a pixel has only four neighbours (4-connectivity), and the raster scan pattern is adopted to scan the binary image.

RASTER SCAN

Raster scanning is a method of visiting or processing image pixels sequentially row by row, from left to right and top to bottom, just like how text is read in English.



Region Labelling

- During the **first** pass, when you encountered an object pixel (*p*), there are three conditions that you need to check:
 - 1) If all four neighbours are **background pixels**, assign a new label to *p*.
 - 2) If only one of the four neighbours is an object pixel that has already received a label, assign its label to *p*.
 - 3) If more than one of the neighbours is an object pixel, check how many of them have received a label.
 - a) If more than one is labelled, assign the smallest label (number) to *p*, and record the equivalences.
 - b) If only one has received a label, assign it to *p*.
- Else:
 - 4) Otherwise (if none of the above is true), then assign a new label to *p*.

IMAGE (0,1)

All neighbors are background → Assign a new label to p

```
[ 0 0 0 ]  
[ 0 p 0 ] ← p = 1, neighbors are all 0
```

Only one labeled neighbor → Assign that label to p

```
[ 0 0 0 ]  
[ 1 p 0 ] ← left is 1 (foreground), already labeled
```

Multiple object neighbors, only one labeled → Assign that label to p

```
[ 0 1 0 ]  
[ 1 p 0 ] ← top = 1 (not labeled), left = 1 (labeled)
```

Multiple neighbors labeled → Assign smallest label and record equivalence

```
[ 1 1 0 ]  
[ 0 p 0 ] ← top-left = 1, top = 1 → both labeled
```

Otherwise → Assign a new label to p

LABELS (1,2,3...)

```
[ 0 0 0 ]  
[ 0 2 0 ] ← p gets new label = 2
```

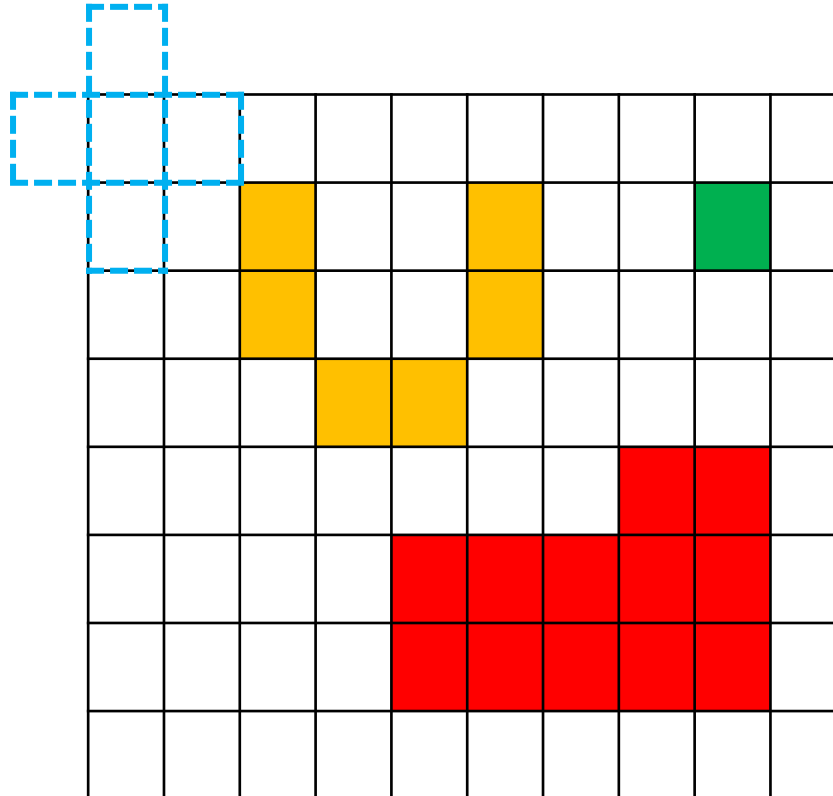
```
[ 0 0 0 ]  
[ 2 2 0 ] ← p gets label = 2 from left
```

```
[ 0 0 0 ]  
[ 2 2 0 ] ← p gets label = 2 from left
```

```
[ 2 3 0 ]  
[ 0 2 0 ] ← p gets min(2,3) = 2, record 2 ≡ 3
```

Region Labelling

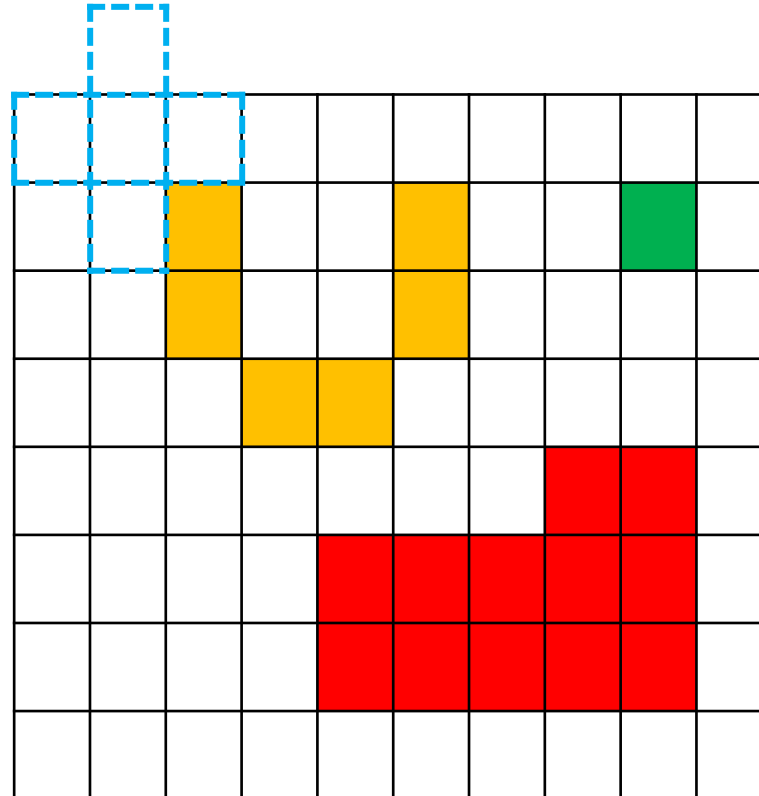
- Start scanning.



Equivalent Table

Region Labelling

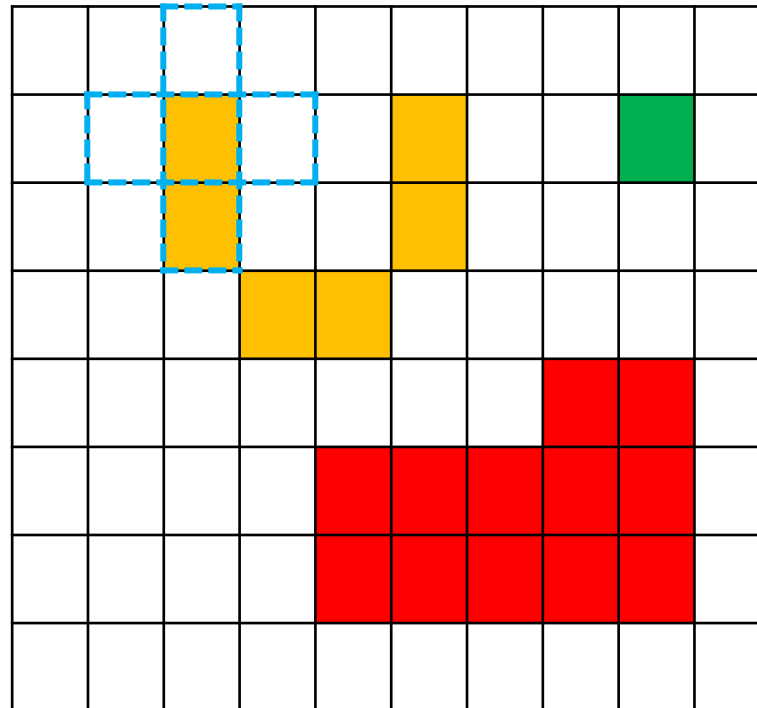
- Continue scanning.



Equivalent Table

Region Labelling

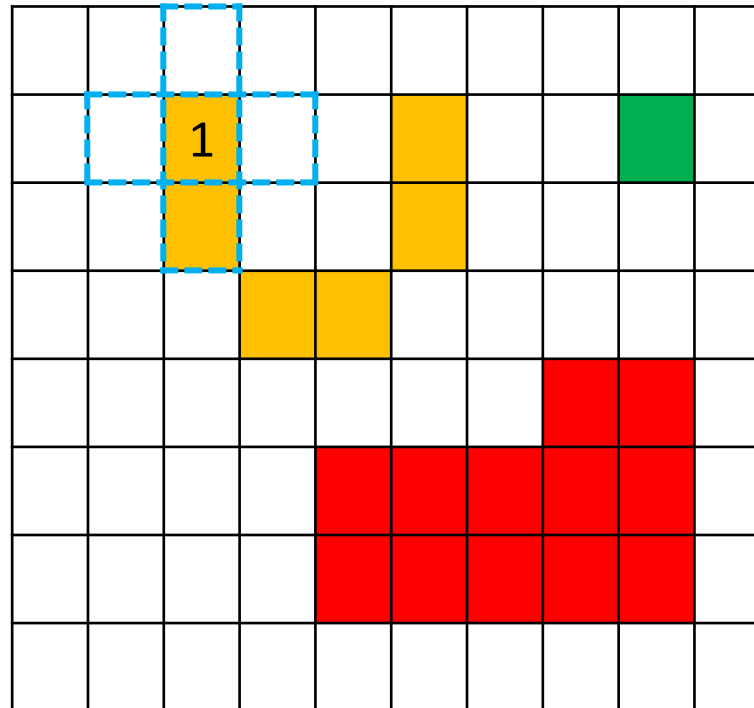
- Although one of the four neighbours is an object pixel, but there is no label that can be assigned to it, so assign a new label to this object pixel, p. - **Condition (4)**



Equivalent Table

Region Labelling

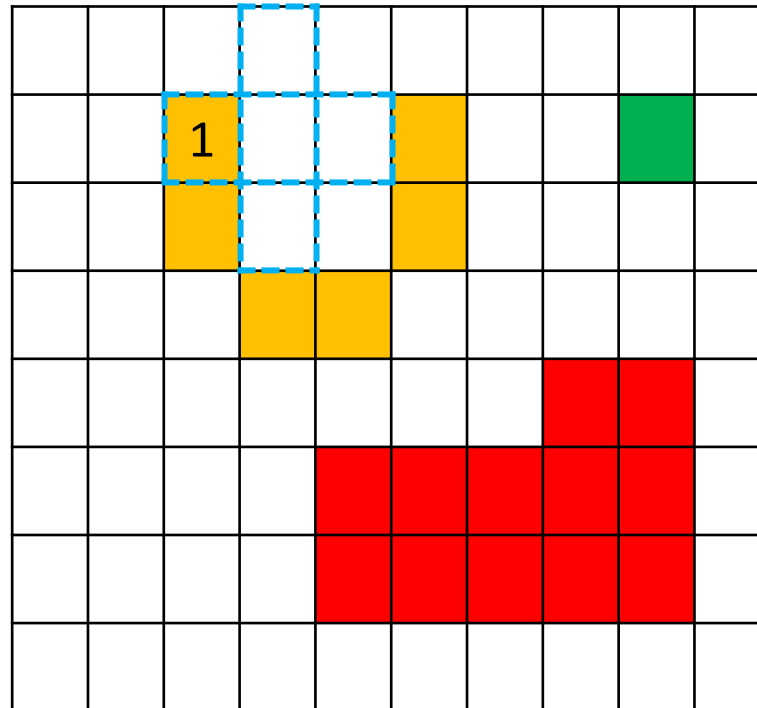
- Although one of the four neighbours is an object pixel, but there is no label that can be assigned to it, so assign a new label to this object pixel, p. - [Condition \(4\)](#)



Equivalent Table

Region Labelling

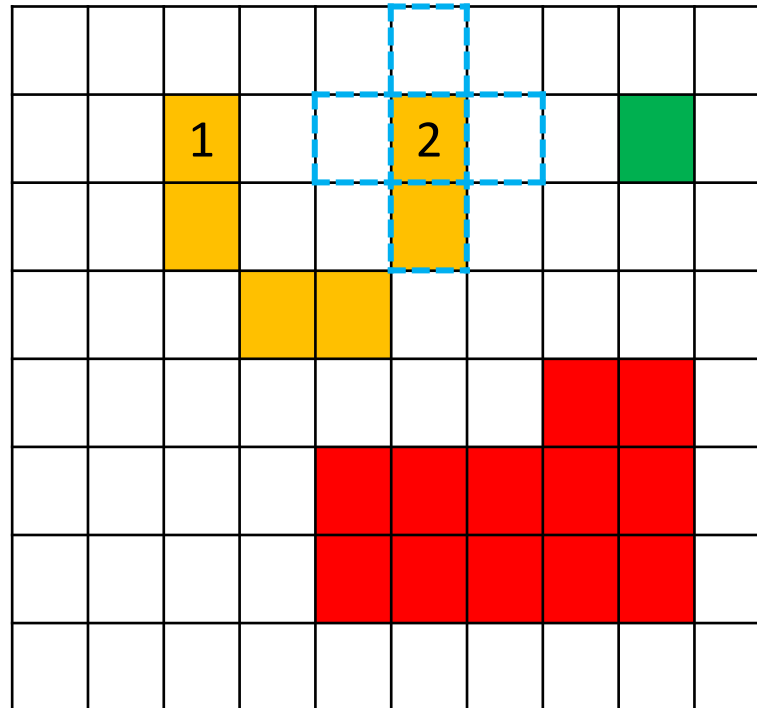
- Continue scanning.



Equivalent Table

Region Labelling

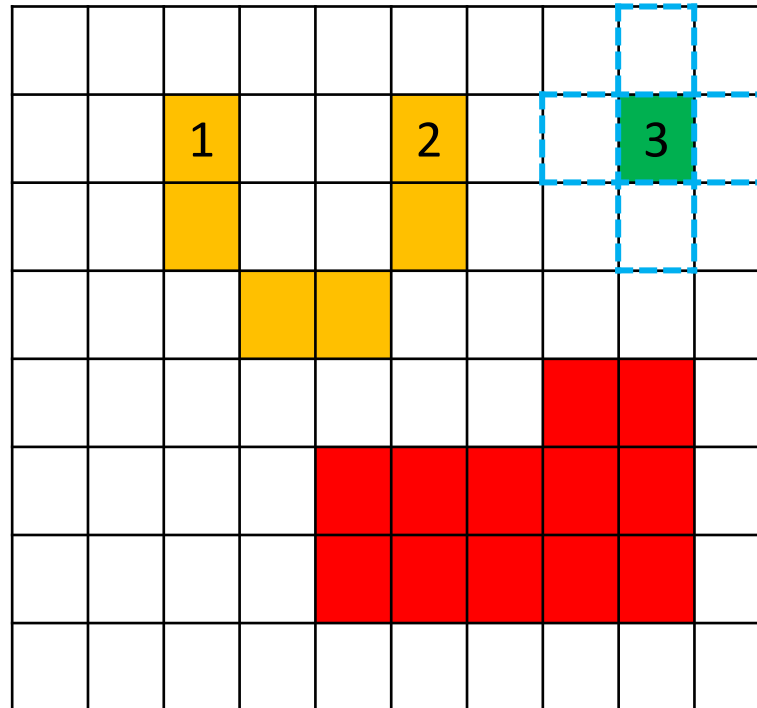
- Although one of the four neighbours is an object pixel, there is no label that can be assigned to it, so assign a new label to this object pixel, p. - Condition (4)



Equivalent Table

Region Labelling

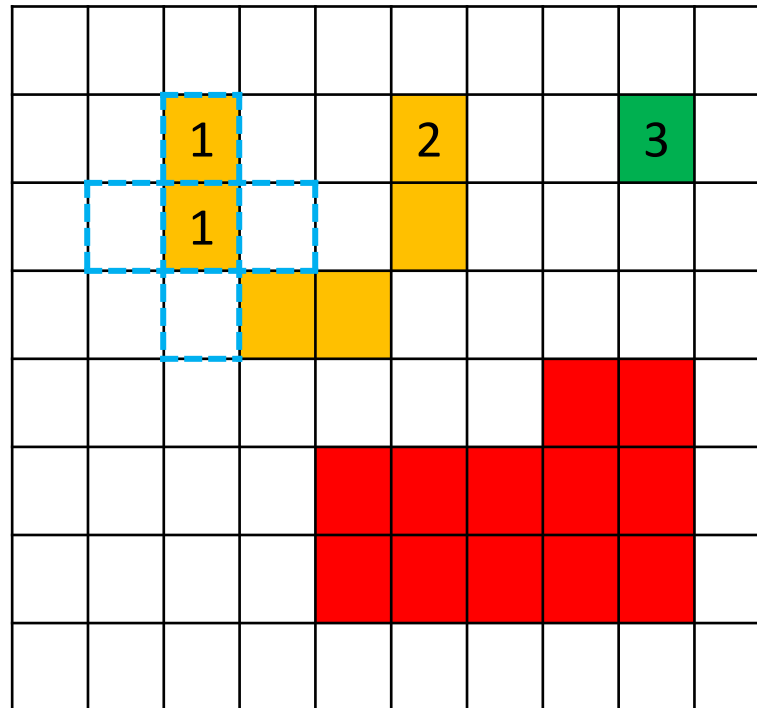
- None of the neighbours is an object pixel, so assign a new label to this object pixel, p. - Condition (1)



Equivalent Table

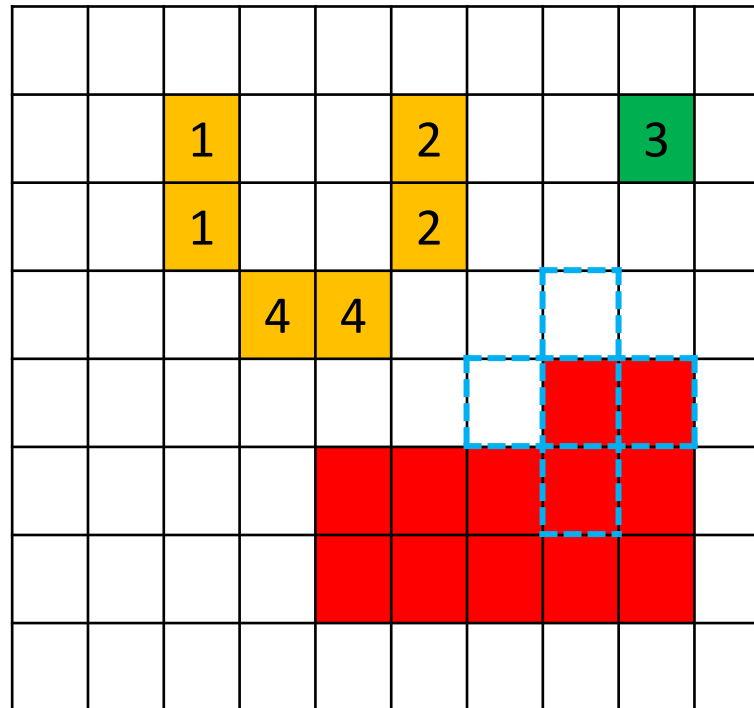
Region Labelling

- One of the four neighbours is an object pixel that has already received a label, assign its label to this object pixel, p. – **Condition (2)**

[illegible]

Region Labelling

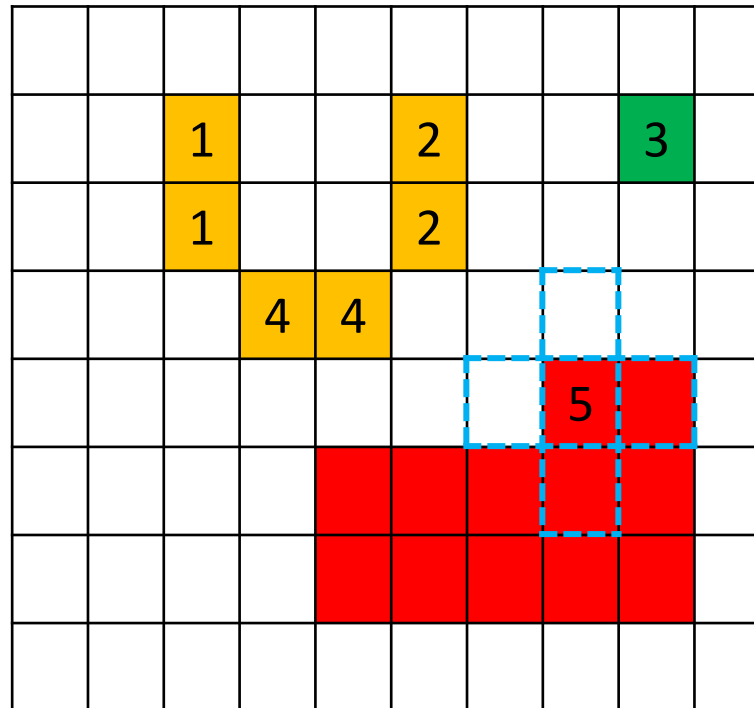
- Continue scanning and assigning labels.



Equivalent Table

Region Labelling

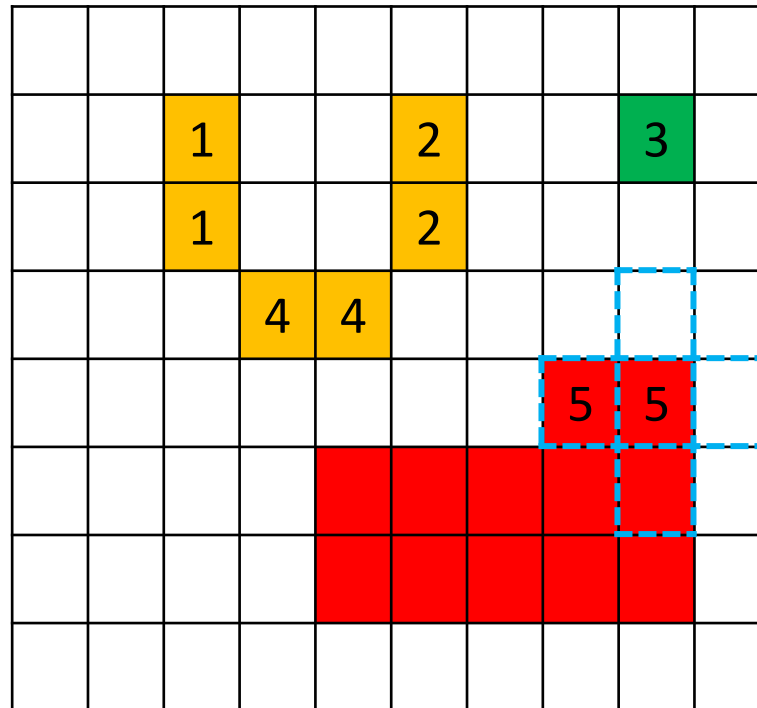
- Although more than one of the neighbours is an object pixel, there is no label that can be assigned to it, so assign a new label to this object pixel, p. – **Condition (4)**



Equivalent Table

Region Labelling

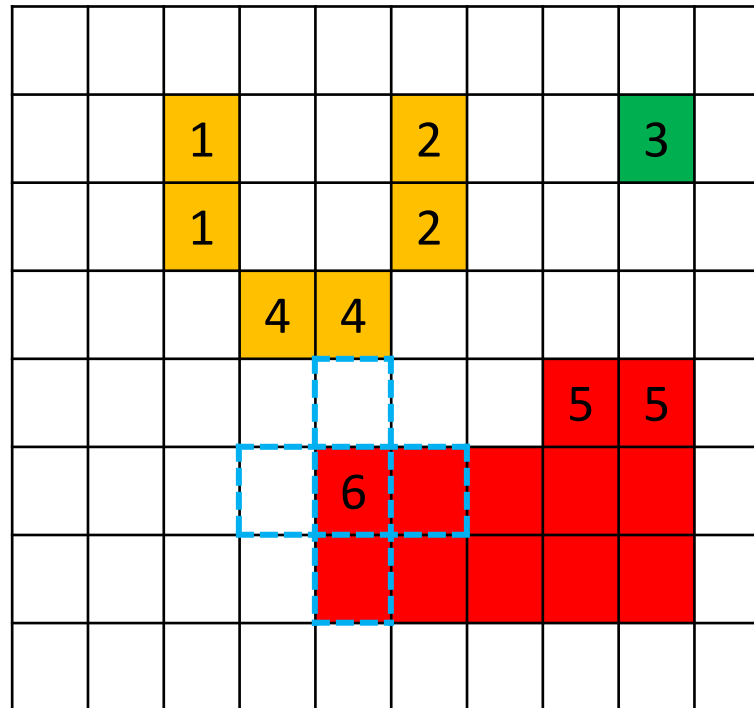
- More than one of the neighbours is an object pixel, and there is only one neighbour with a label, so assign this label to the object pixel, p. – [Condition \(2\)](#)



Equivalent Table

Region Labelling

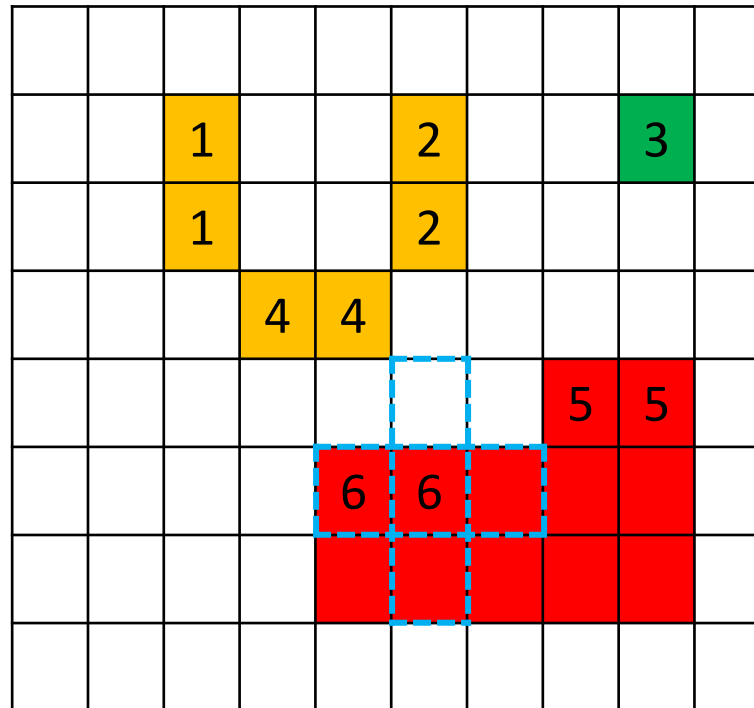
- Although more than one of the neighbours is an object pixel, there is no label that can be assigned to it, so assign a new label to this object pixel, p. – **Condition (4)**



Equivalent Table

Region Labelling

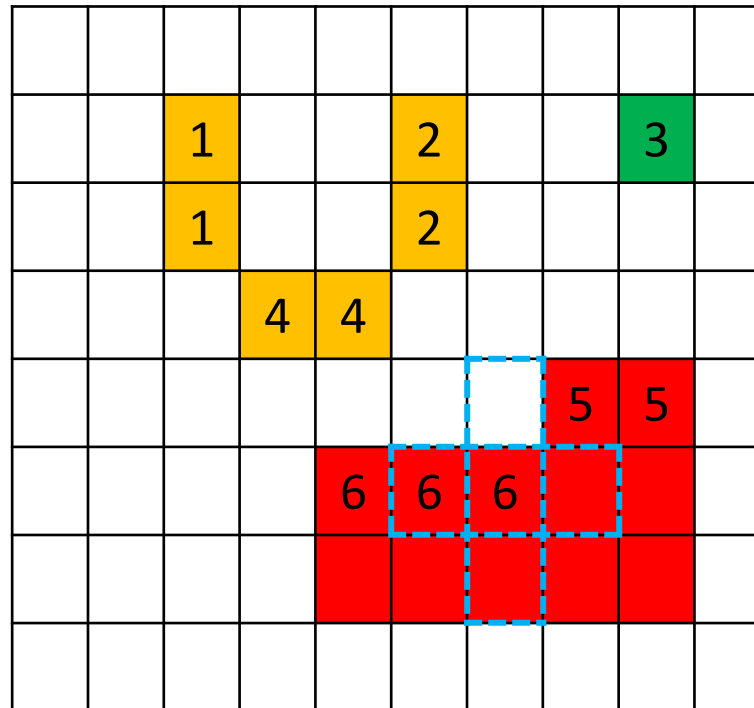
- More than one of the neighbours is an object pixel and there is only one neighbour with a label, so assign this label to this object pixel, p. – [Condition \(2\)](#)



Equivalent Table

Region Labelling

- More than one of the neighbours is an object pixel and there is only one neighbour with a label, so assign this label to this object pixel, p. – **Condition (2)**

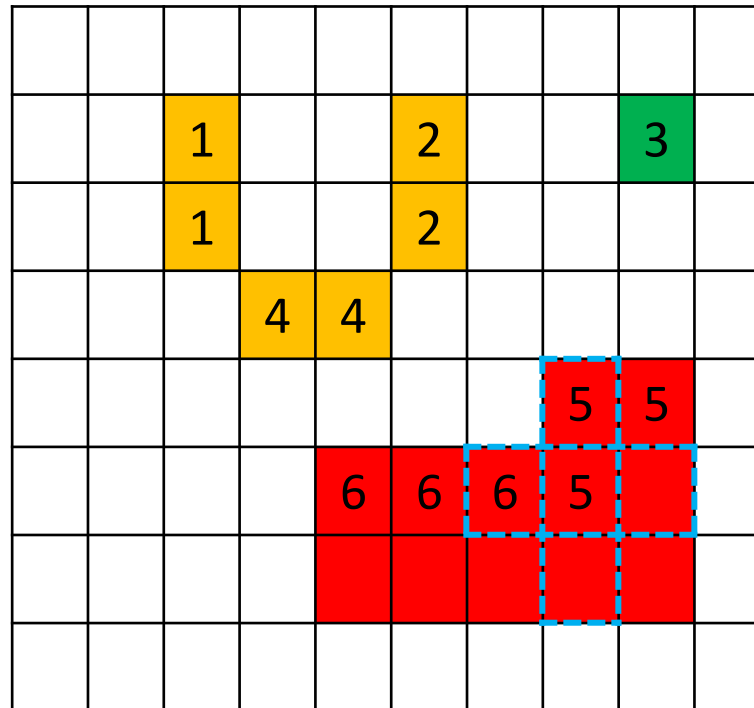


Equivalent Table

Region Labelling

- More than one of the neighbours is an object pixel and there are two labelled neighbours, so assign the smallest label to this object pixel, p, and record the equivalences in a table. —

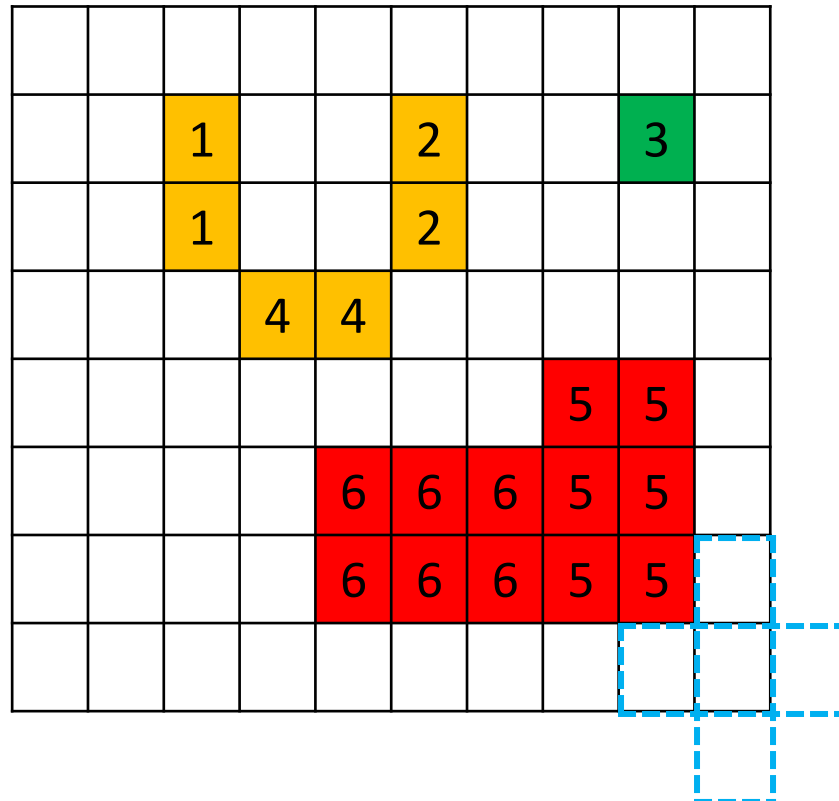
Condition (3)



Equivalent Table
5 = 6

Region Labelling

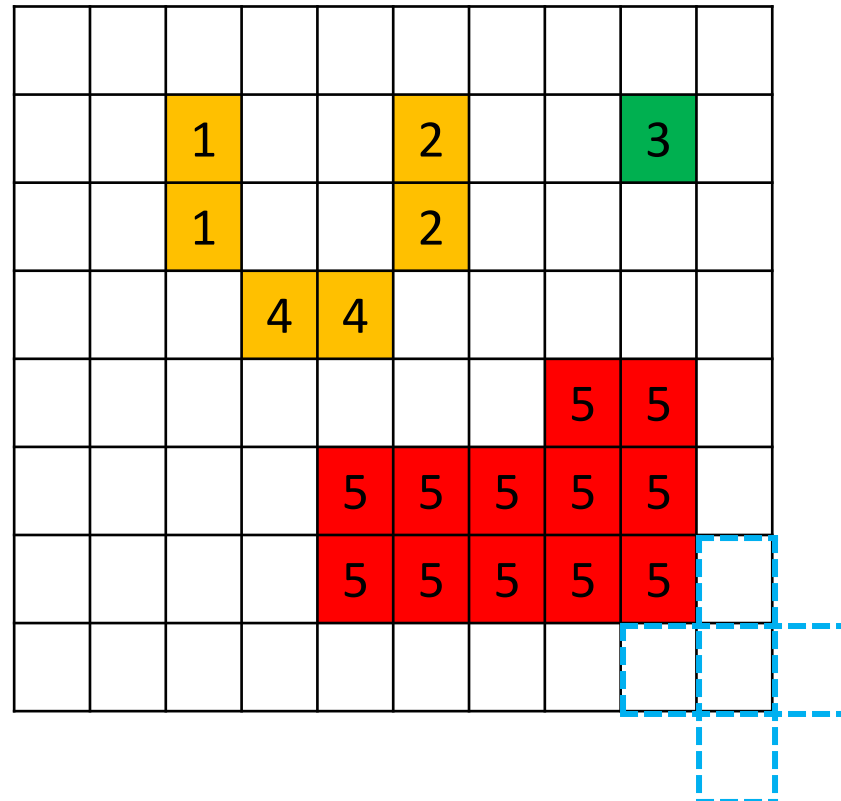
- Continue scanning labelling until the end.



Equivalent Table
5 = 6

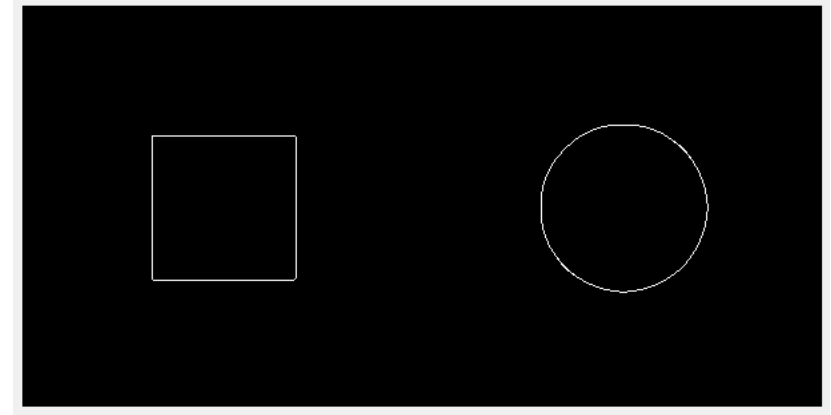
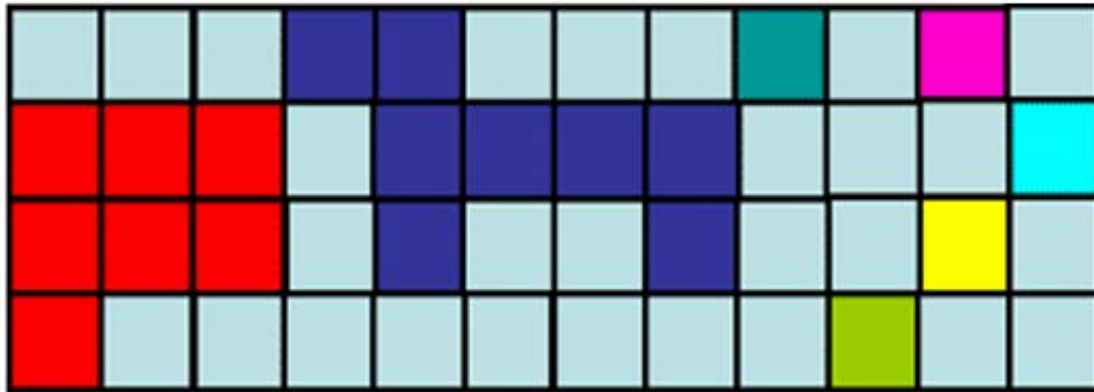
Region Labelling

- During the **second pass**, check all labels and update labels with larger values to their corresponding smaller values based on the equivalences in the table.



Equivalent Table
5 = 6

Once an image is **segmented** (e.g., using thresholding, region growing, or watershed), the next step is to **represent** the segmented regions in a meaningful and compact way to enable **recognition**, **classification**, or **matching**.



- ❖ How many objects are there? -> Region Labelling
- ❖ What are these objects?

What's Next

To **represent** and **describe** the segmented region.

- ❖ We can use either its **internal characteristics** (e.g., the pixels within the region) or **external characteristics** (e.g., the region's boundary).
- ❖ Once we determine which type of characteristic to use for representation, the next step is to **describe** the region based on the chosen approach.

Representation

Representation refers to the process of converting processed image data into a form suitable for further analysis or interpretation. It deals with how to store or express the region or boundary (e.g., boundary coordinates, region pixels, etc.).

PURPOSE

- ❖ To highlight meaningful structures or features (e.g., shapes, boundaries, regions).
- ❖ To reduce complexity while preserving essential information.

TYPES OF REPRESENTATION:

- ❖ Boundary Representation (used for shape-based analysis)
- ❖ Region Representation (used for texture- or area-based analysis)

Description (Feature Extraction)

Description is the process of extracting **quantitative information** (called **features**) from the **representation** of an image.

PURPOSE

- ❖ To provide a compact, discriminative, and informative set of values for further tasks like classification, recognition, or matching.

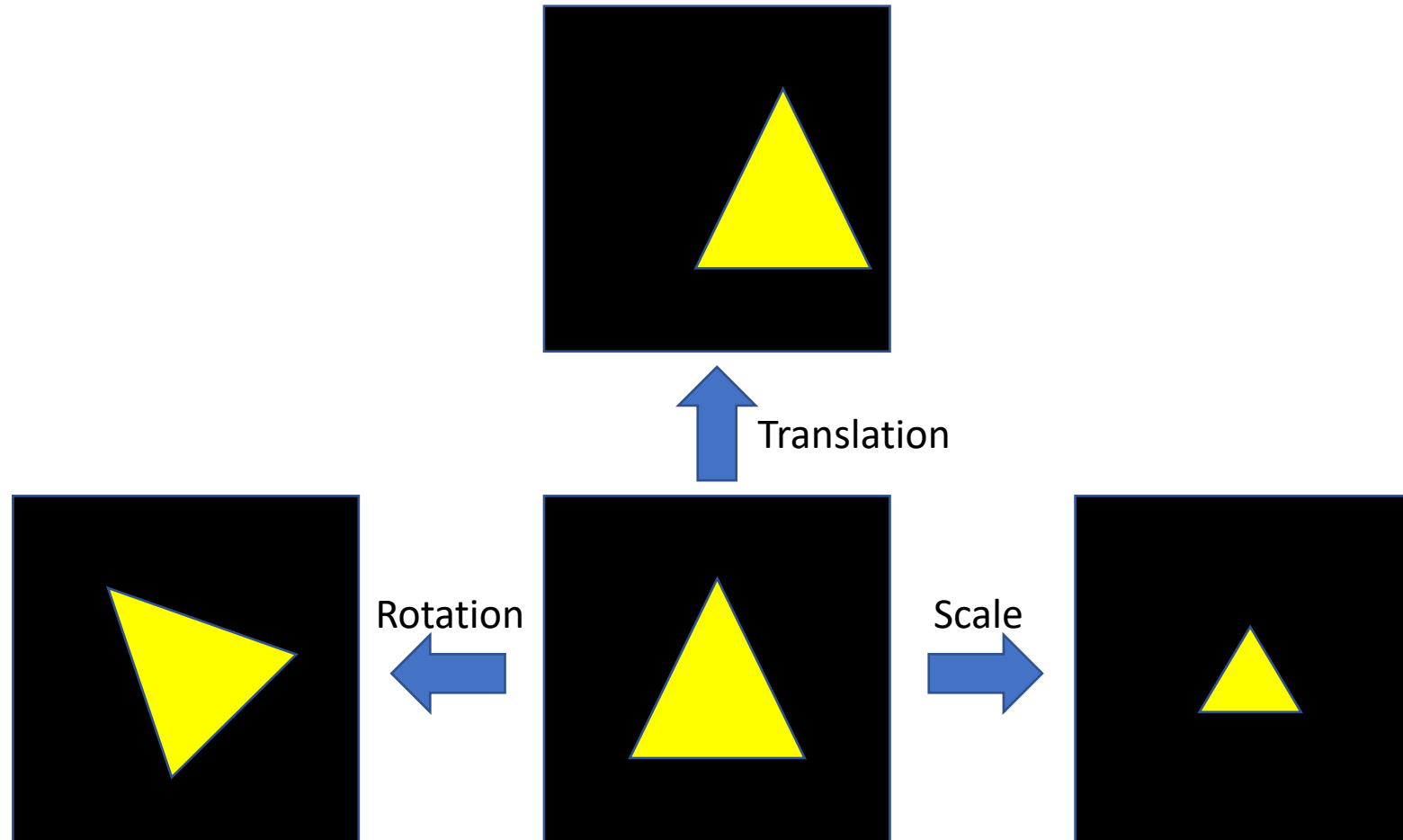
COMMON FEATURES

- ❖ Shape features: perimeter, area, aspect ratio, compactness.
- ❖ Texture features: coarseness, contrast, directionality (e.g., GLCM – Gray-Level Co-occurrence Matrix).
- ❖ Color features: histograms in RGB, HSV space.
- ❖ Topological features: connectivity, number of holes.

Representation and Description

- ❖ Representation is needed to isolate or select specific regions before further processing.
- ❖ Descriptors (numerical values) are used to describe these regions for purposes such as:
 - ✓ Comparison
 - ✓ Selection
 - ✓ Filtering
- ❖ Example: Using a descriptor to find regions with a perimeter longer than a predefined threshold.
- ❖ The description should be invariant to transformations like:
 - ✓ Rotation
 - ✓ Translation
 - ✓ Scale changes
- ❖ This invariance ensures consistent identification of the same object, whether in:
 - ✓ The same image
 - ✓ Different images

Representation and Description



Boundary Representation

Assuming we've chosen to work with the boundary of a region, the next step is to generate its representation in a way that the computer can process and understand. To achieve this, we can use techniques such as:

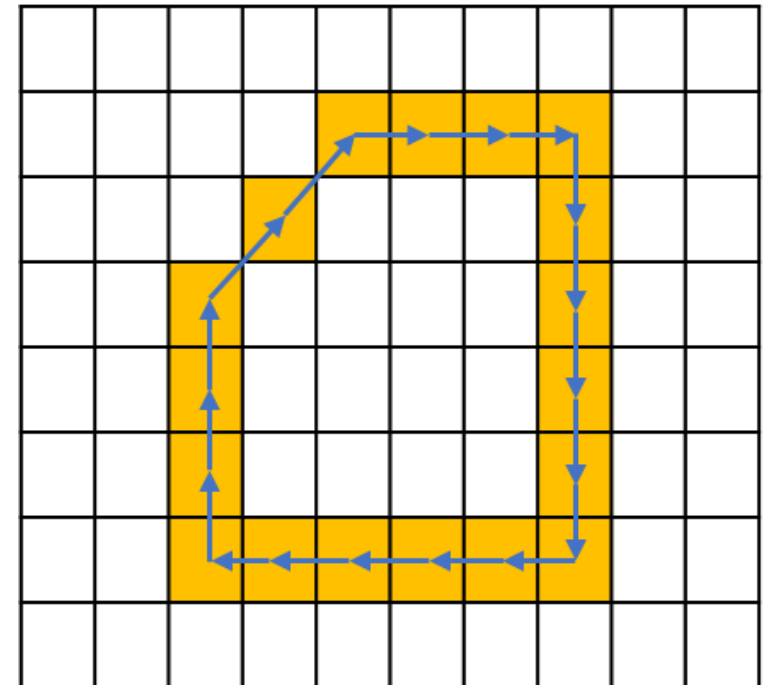
- ❖ Chain Code
- ❖ 1D Signature

These methods allow us to translate the boundary into a format that is both computationally efficient and robust for further analysis.

Chain Code

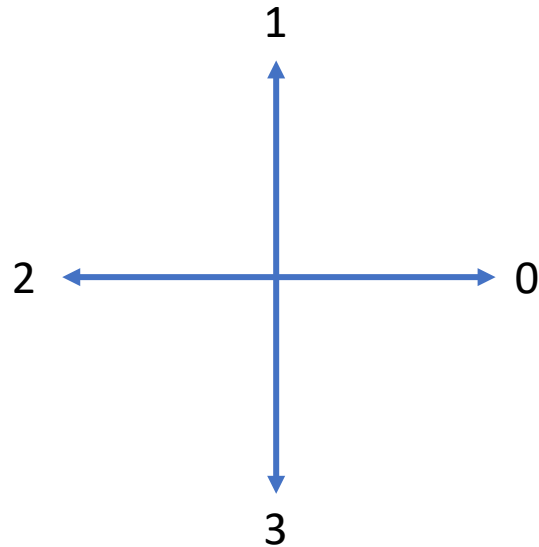
Chain Code is a method used to **represent the boundary** of a region in an image by **describing the contour** as **a sequence of directional moves**. Each move in the sequence indicates a change in direction as you follow the boundary of an object, making it a compact and efficient representation of the object's shape.

- ❖ Treat the boundary as a sequence of connected pixels.
- ❖ Code the directions by using a set of numbers.

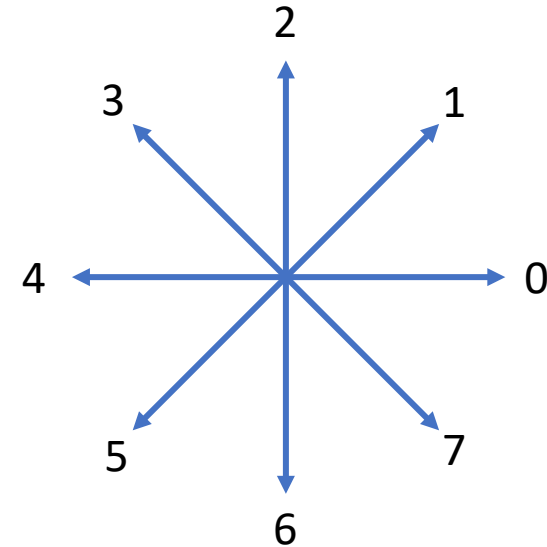


Chain Code

- ❖ The chain code can be derived using either 4-connectivity or 8-connectivity schemes.
- ❖ The resulting chain code, regardless of the connectivity method used, is commonly referred to as the **Freeman Chain Code**.



4-Connectivity

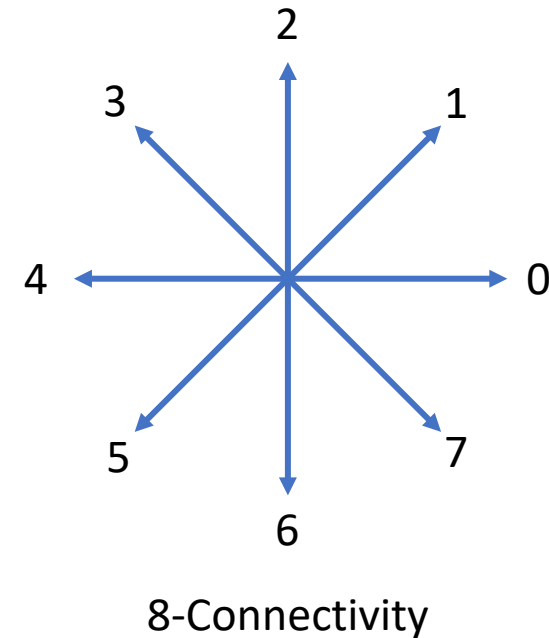
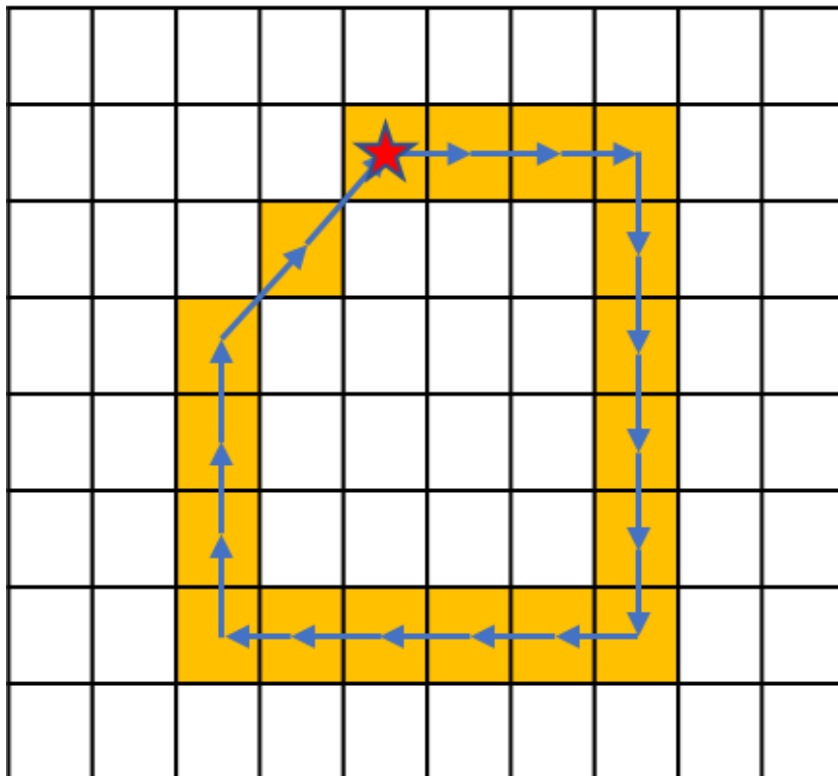


8-Connectivity

Chain Code

- ❖ If the pixel labelled with a star is used as the starting point, then the chain code for this is:

0 0 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1

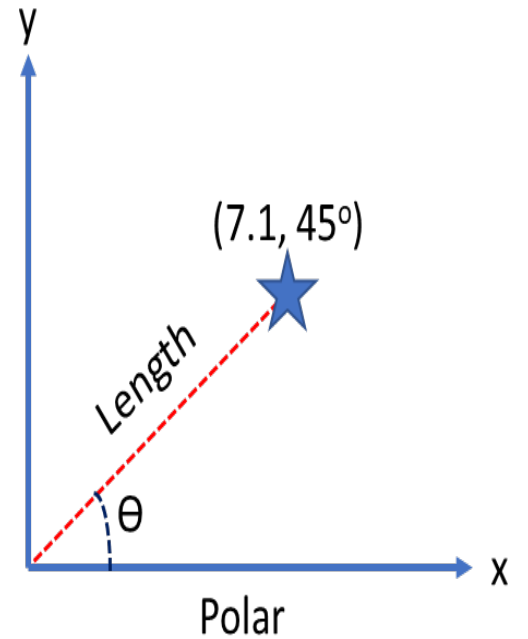
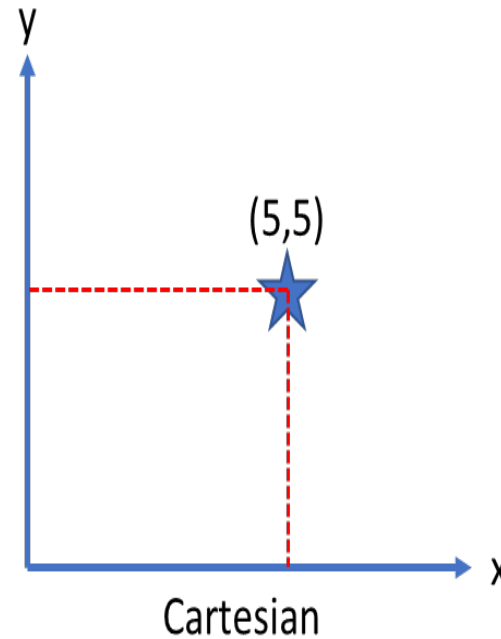


1D signature

The goal of this approach is to represent the **boundary** of a region in a **1D** form that can help in **identifying the shape**. The key idea is to use **polar coordinates** to describe the **distance** between the **centroid of the region** and the **boundary at various angles**.

STEPS

1. Locate the Centroid of the Region.
2. Convert Cartesian Coordinates to Polar Coordinates.
3. Measure the Distance Between the Centroid and the Boundary.
4. Output: The resulting 1D signature is a list of distances corresponding to different angles.



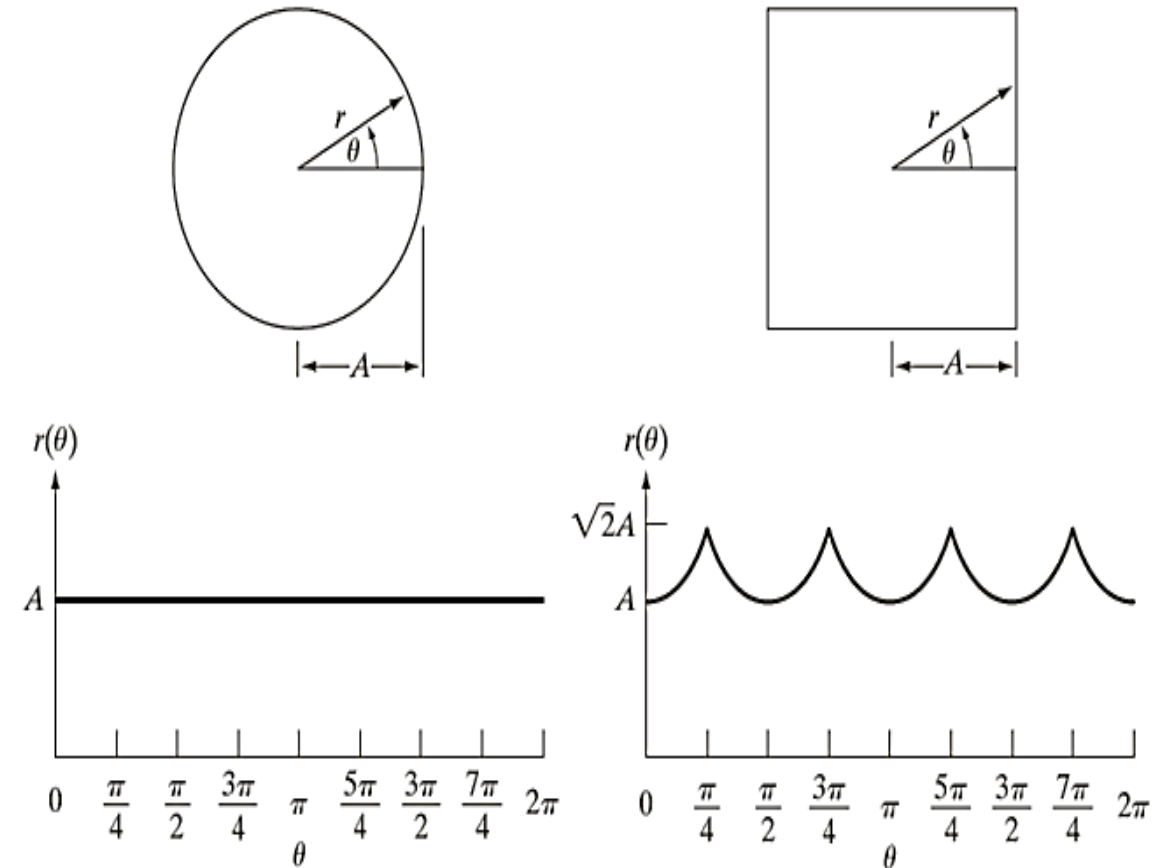
Interpretation of the 1D Signature

The 1D Signature captures the profile of the shape's boundary in a compact form.

- ❖ Changes in distances (the radial values) reveal important information about the shape's curvature, indentations, and outlines.
- ❖ Shape Identification: Different shapes will have distinct patterns of distances as the angle varies.

EXAMPLE

- ❖ A circle will have a constant distance at all angles.
- ❖ A triangle will have peaks in the distance at the vertices and flat regions along the edges.
- ❖ An ellipse will show varying distances along its major and minor axes.



Source: R. C. Gonzalez and R. E. Woods, Digital Image Processing, Pearson, 2018

Description

Once the boundary representation is obtained, we can generate its description using various techniques, such as:

- ✓ Perimeter
- ✓ Shape Number
- ✓ Fourier Descriptors
- ✓ Corners

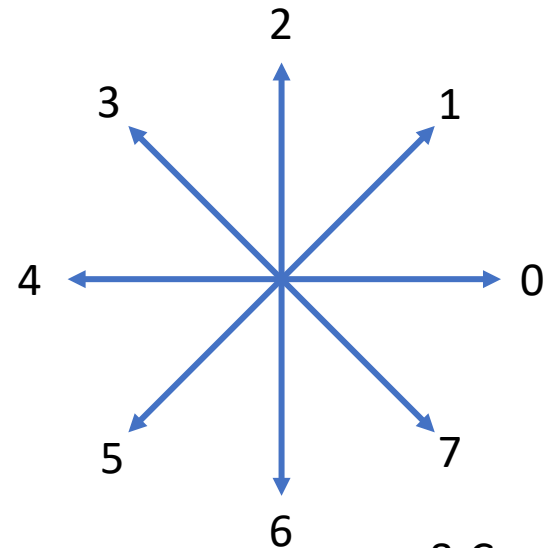
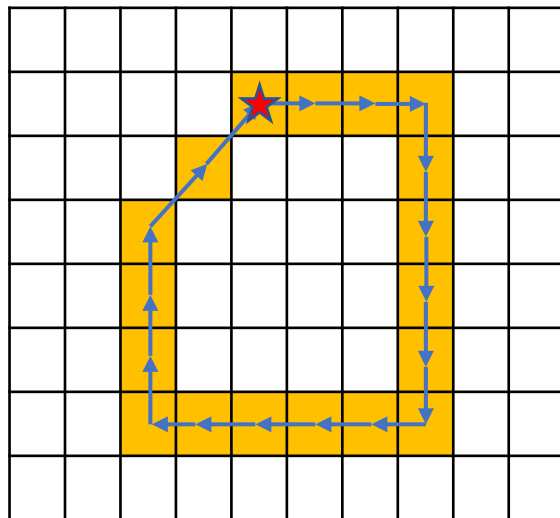
Shape Number

The Shape Number is a numerical representation used to describe the **overall shape** of a **contour** or **object** in an image. It is a **topological descriptor** that can be used to **classify** and **compare shapes**, especially when dealing with **simple, closed shapes** like **polygons**.

STEPS

- ❖ We will use back the same example for demonstration purpose

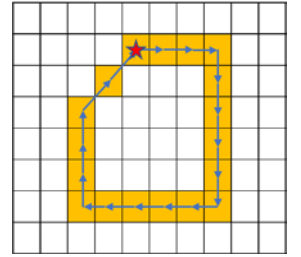
0 0 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1



8-Connectivity

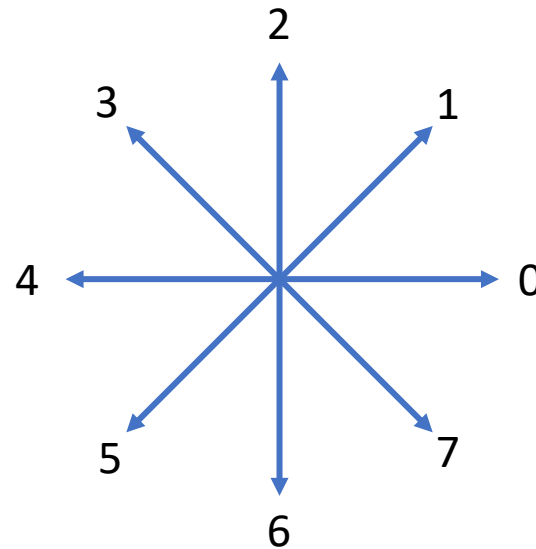
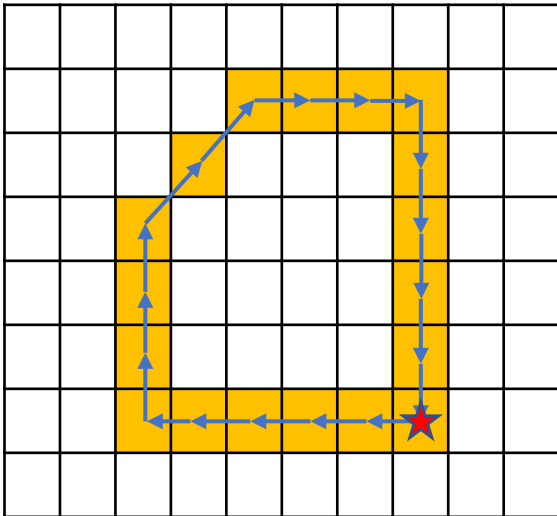
Shape Number

❖ “Issue” with chain code, what if a different starting point is selected?



0 0 0 6 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1

4 4 4 4 4 2 2 2 1 1 0 0 0 6 6 6 6 6

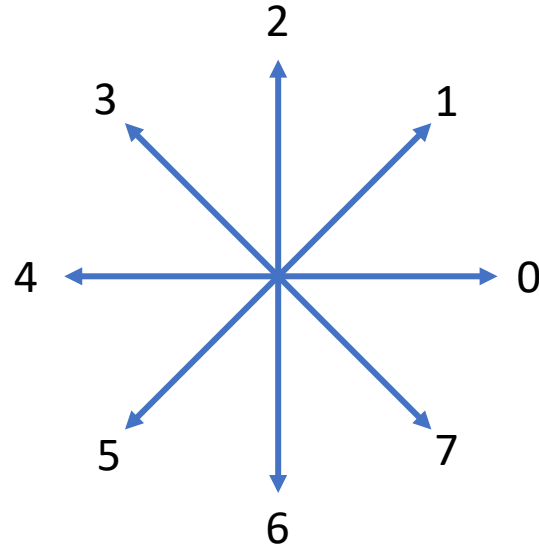
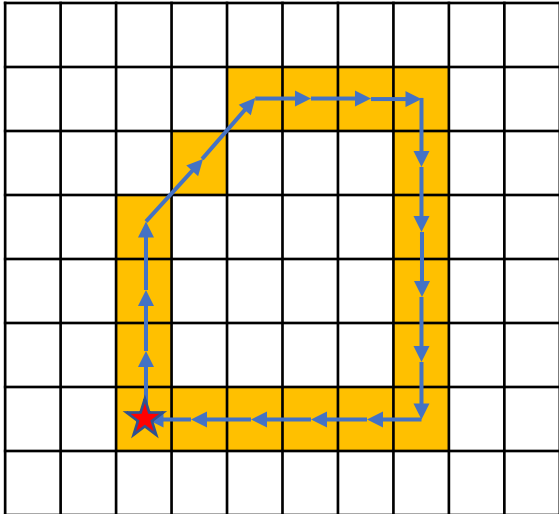


8-Connectivity

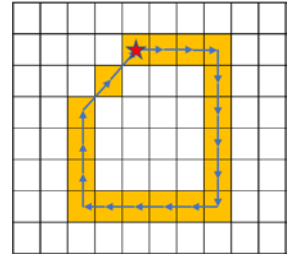
Shape Number

❖ What if another different starting point is selected?

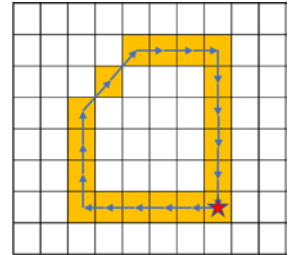
2 2 2 1 1 0 0 0 6 6 6 6 6 4 4 4 4 4



8-Connectivity



0 0 0 6 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1



4 4 4 4 4 2 2 2 1 1 0 0 0 6 6 6 6 6

Shape Number

- ❖ We can normalize the chain code with respect to the starting point by:
 - a) Treat the chain code as a circular sequence of numbers.
 - b) Rotate the numbers until the numbers form an integer with smallest magnitude.

→ 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1 0 0 0

Magnitude: 66666444422211000

→ 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1 0 0

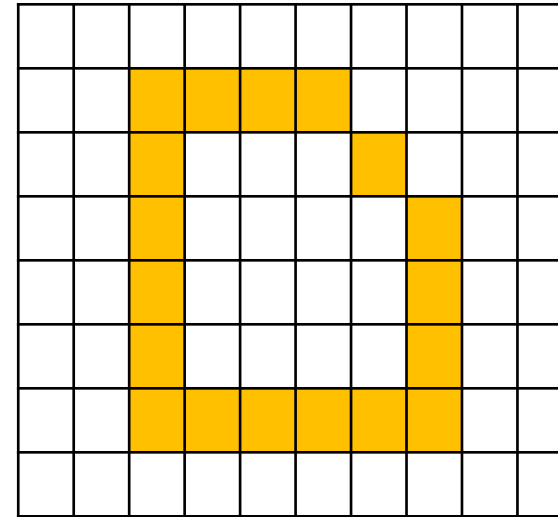
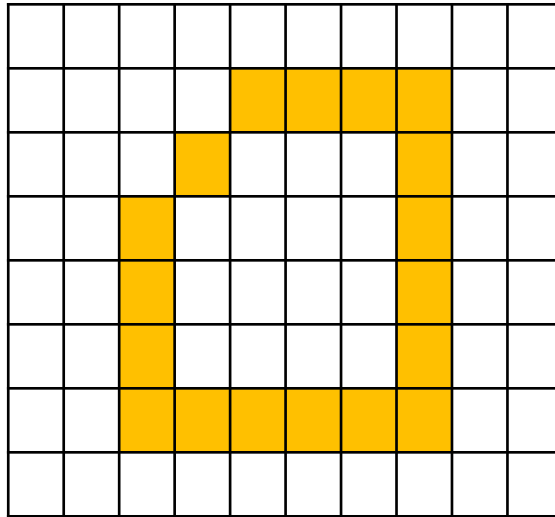
Magnitude: 06666644442221100

→ 0 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1 0

Magnitude: 00666664444222110

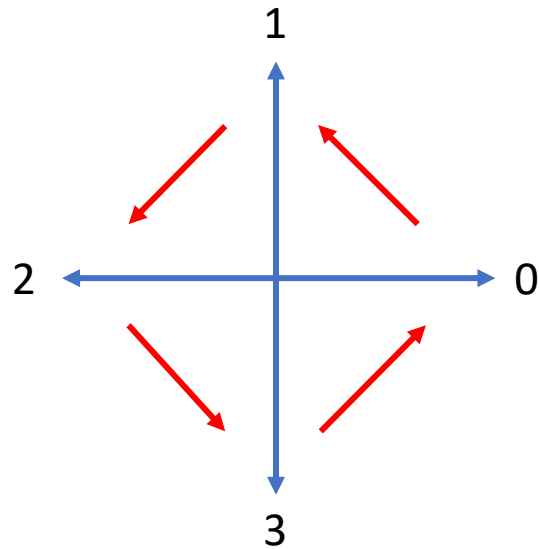
Shape Number

- ❖ This solved one of the issues (when different starting points are selected), but what if the boundary is rotated?

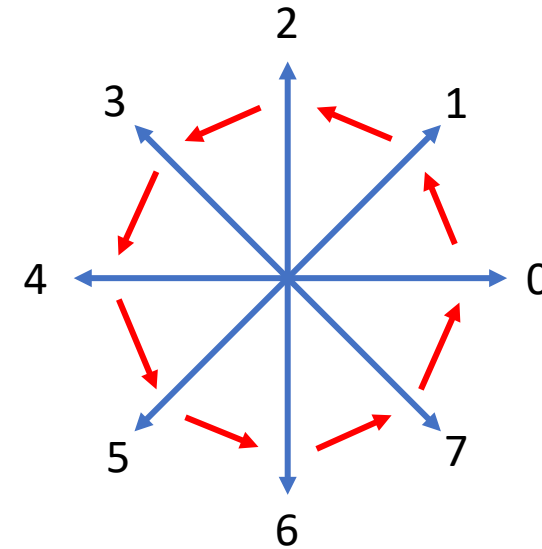


Shape Number

- ❖ We can normalize the chain code with respect to rotation by finding the difference of the chain code.
- ❖ In this case, the difference is referring to the number of direction changes (in counter-clockwise).



4-Connectivity

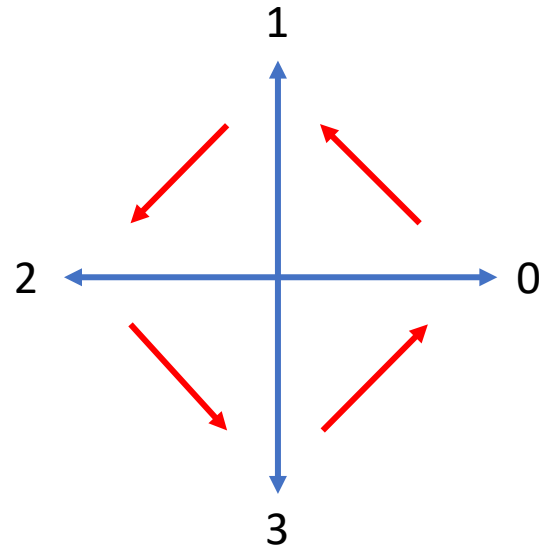


8-Connectivity

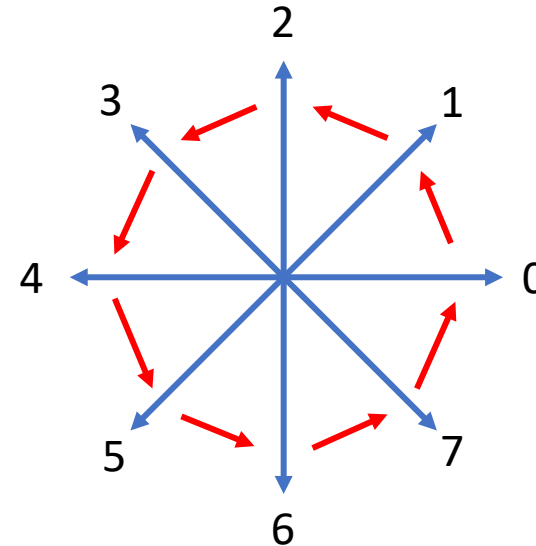
Shape Number

For example:

- ❖ From 0 to 3, there are 3 changes.
- ❖ From 3 to 1, there are 2 changes.



4-Connectivity



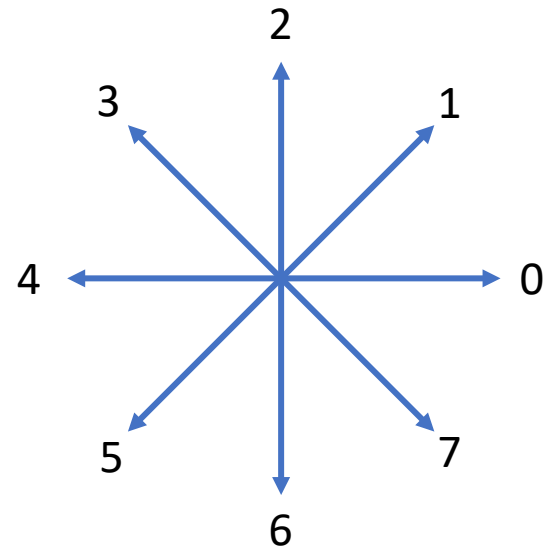
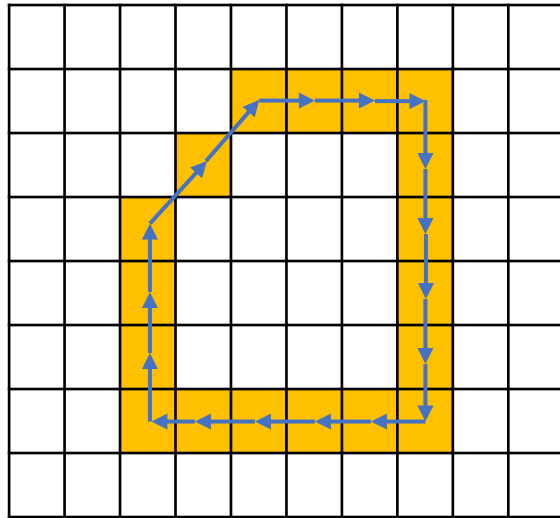
8-Connectivity

Shape Number

- ❖ The difference is defined as the changes between the adjacent numbers (in this case we used back the same example):

Chain Code: 0 0 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1

Difference: 0 0 6 0 0 0 0 6 0 0 0 0 6 0 0 7 0

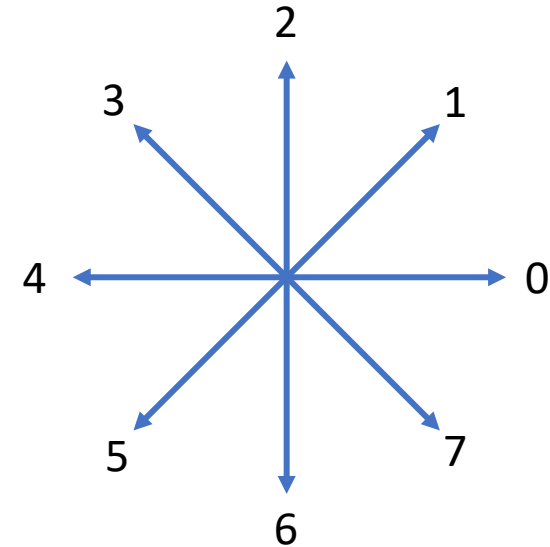
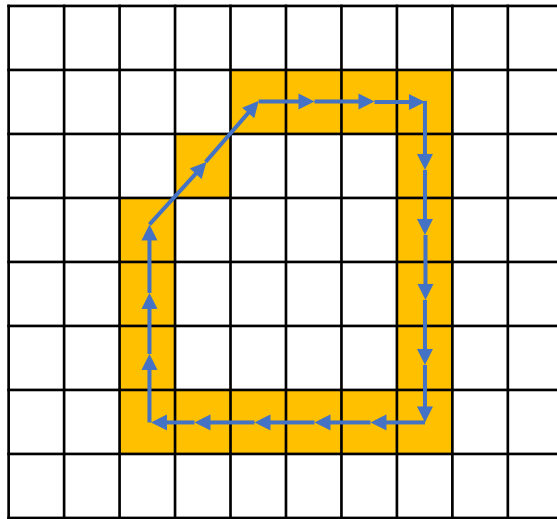


8-Connectivity

Shape Number

- ❖ For example, from 0 to 6, there are 6 direction changes.
- ❖ Hence, the output is 6. Same goes to 6 to 4.

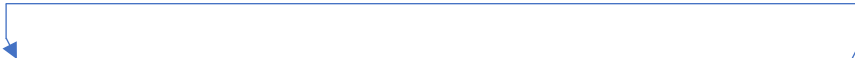
Chain Code: 0 0 0 6 6 6 6 6 4 4 4 4 2 2 2 1 1
Difference: 0 0 6 0 0 0 0 6 0 0 0 0 6 0 0 7 0



8-Connectivity

Shape Number

- ❖ The final step is to calculate the difference between the last digit and the first digit of the chain code.
- ❖ Then, rearrange the sequence to obtain the smallest possible magnitude.
- ❖ It's important to treat the chain code as a circular sequence of numbers during this process.



Chain Code: 0 0 0 6 6 6 6 6 4 4 4 4 4 2 2 2 1 1

Difference: 7 0 0 6 0 0 0 0 6 0 0 0 0 6 0 0 7 0

Shape No: 0 0 0 0 6 0 0 0 0 6 0 0 7 0 7 0 0 6

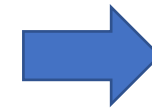
Fourier Descriptor

The Fourier transform can be used to transform boundary points (coordinates) of a shape into what are known as Fourier descriptors. These descriptors provide a compact representation of the shape's boundary, reducing the complexity of the original data while retaining essential features.

STEPS

1. Get Boundary Coordinates (Pixel Positions) of the object
2. Convert the Coordinates into Complex Numbers ($x+iy$)
3. Apply the Discrete Fourier Transform (DFT) to the series
4. Fourier Descriptors - The resulting Fourier descriptors are the complex numbers produced by the DFT.

0	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

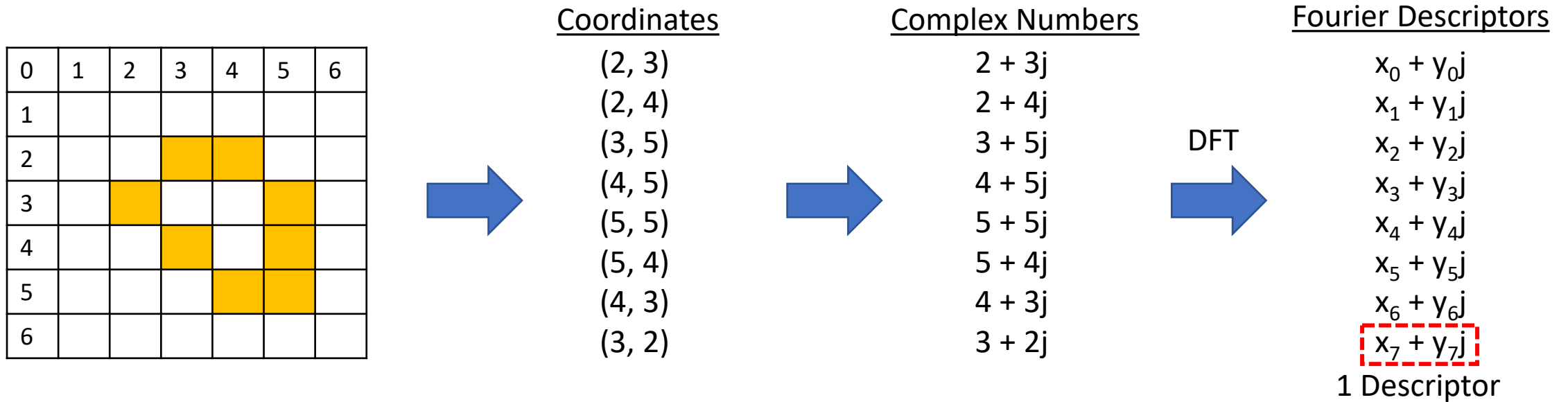


Coordinates

(2, 3)
(2, 4)
(3, 5)
(4, 5)
(5, 5)
(5, 4)
(4, 3)
(3, 2)

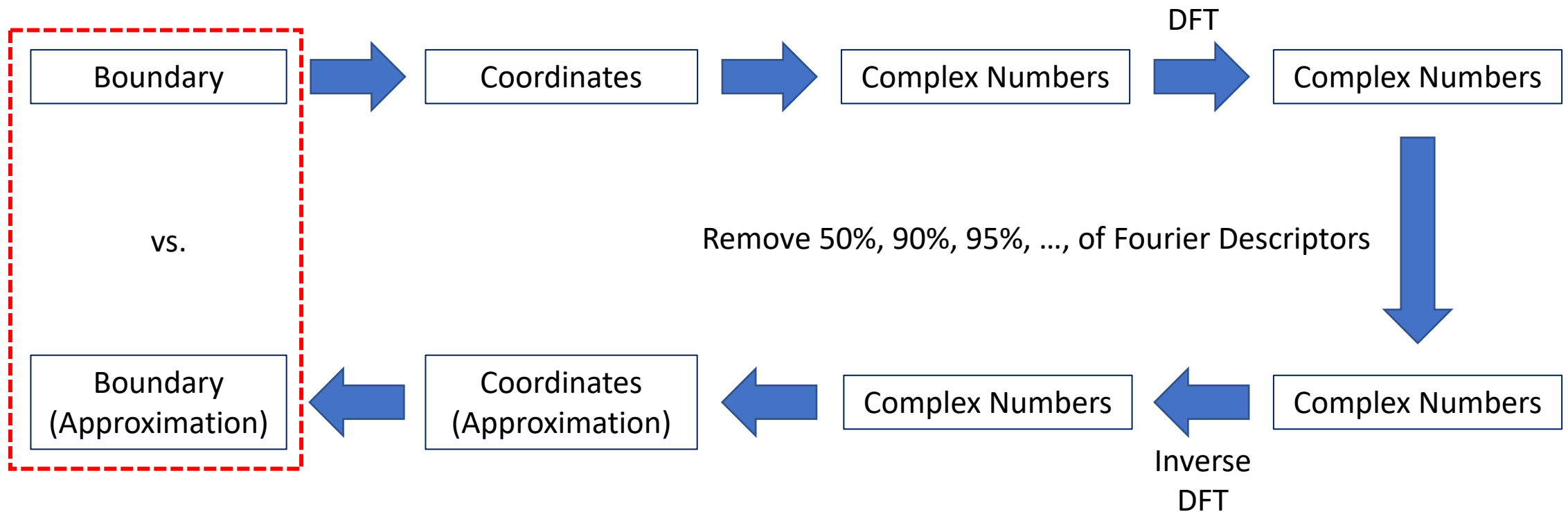
Fourier Descriptor

- ❖ Before we can transform the points using Discrete Fourier Transform (DFT), we need to first change those points into complex numbers.
- ❖ Each point (row, col) will form a pair of complex number (row + j col).

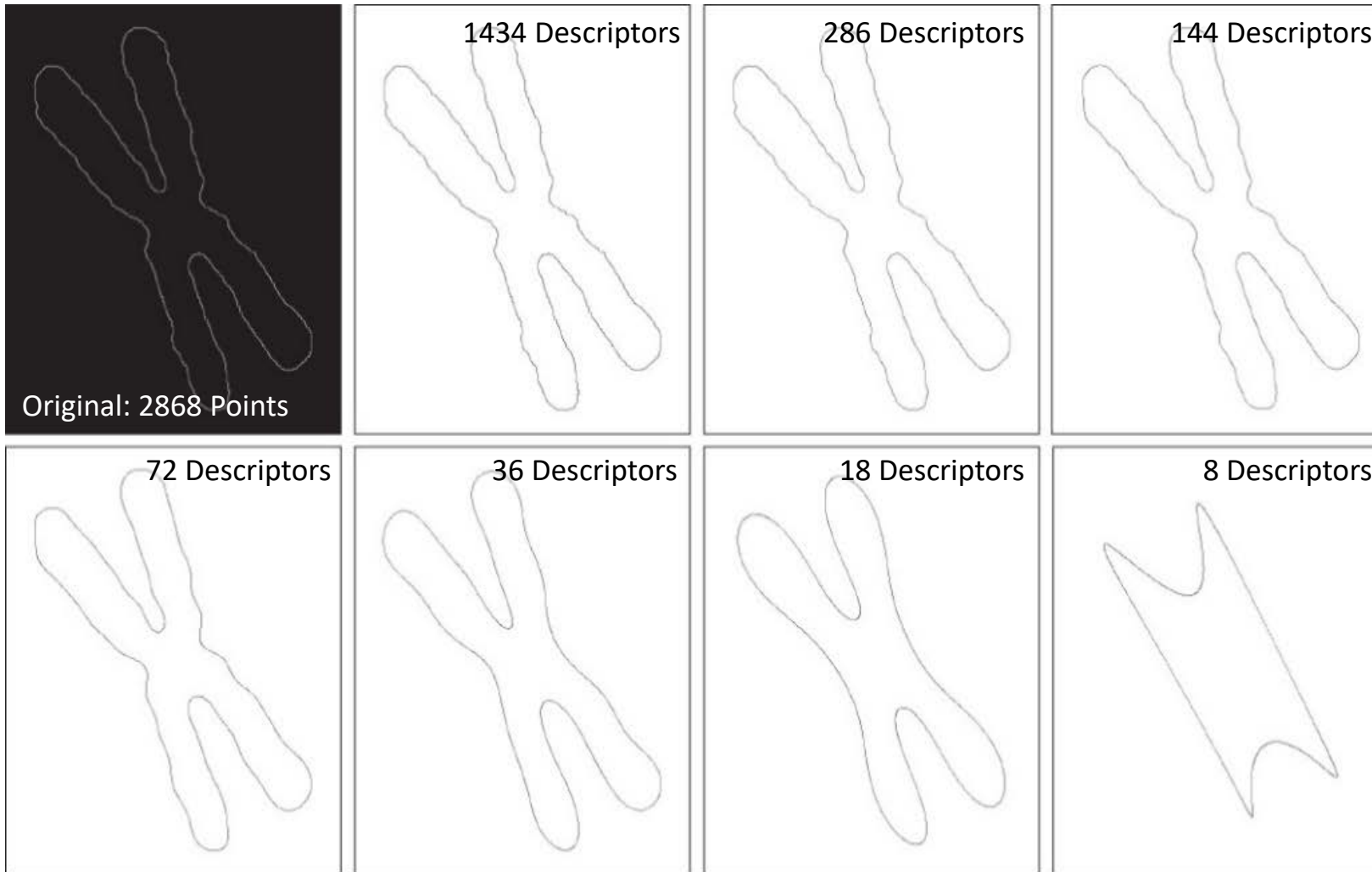


Fourier Descriptor

- ❖ In the following example, we will try to observe the effects of removing those high frequency components.



Fourier Descriptor

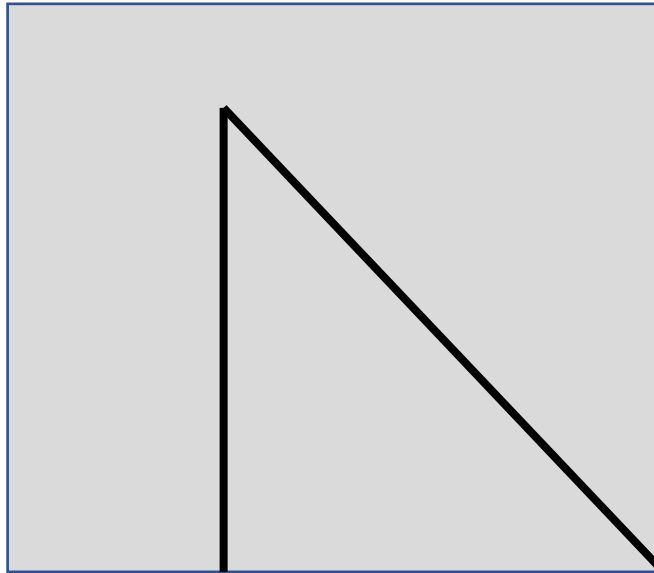


- ❖ We do not have to “save” so many points when using Fourier descriptors.
- ❖ For example, with only 10% of the Fourier descriptors, we are still able to get a very good approximation of the original shape.

Corners

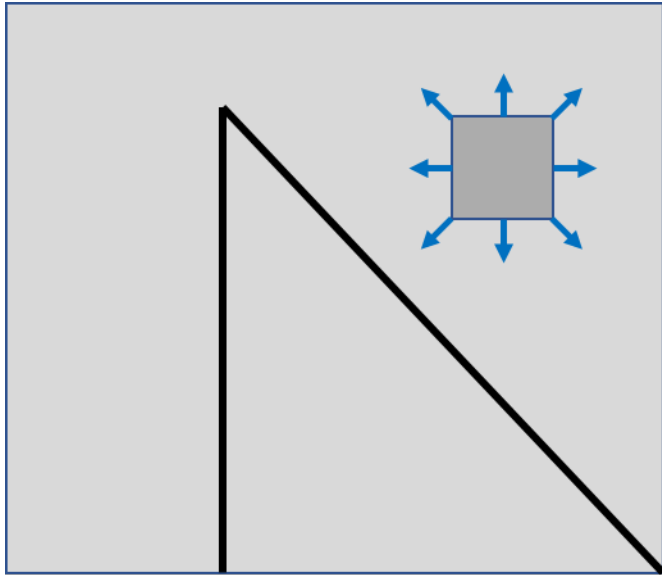
Corners refer to points in an image where the **intensity of the pixel** values **changes sharply** in **multiple directions**. These are considered important features because they are **stable** and **distinctive**, which makes them useful for tasks like **object recognition**, **tracking**, and **image matching**.

- ❖ Given the following image, how would you determine whether there is a corner in the image? and the location of it?

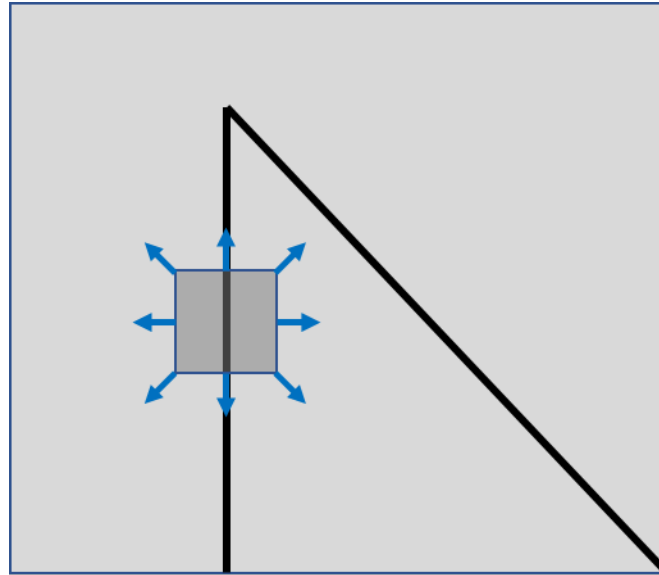


Corners

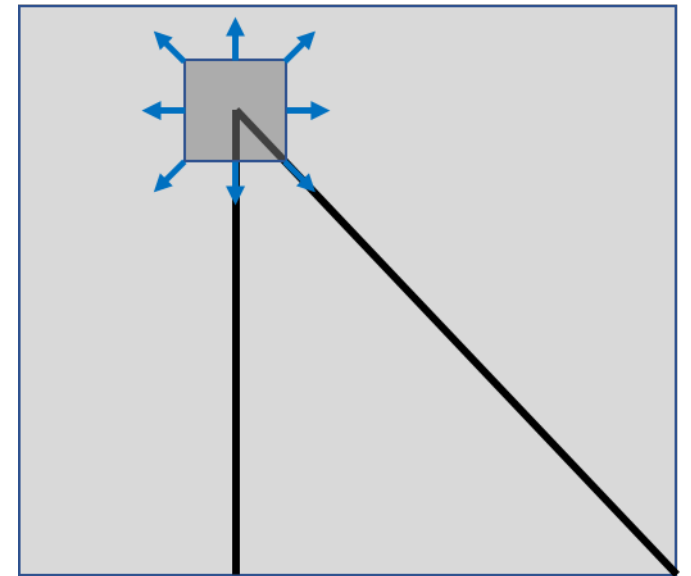
- ❖ The general idea is to use a small window to scan through the image and then shift the small window in all the directions at every location to see if there is any significant intensity changes in all the directions.



For flat region, there is no significant changes no matter which direction we shift the window.



For edge, there is no significant changes along the edge direction (that means no changes when we shift the window up and down).



For corner, there will be significant changes in all the directions.

Corners

- ❖ But what does “significant” mean?
 - ❖ It depends on the threshold (or sensitivity) value you have set.
 - ❖ You can adjust this value to get the desired output.
- ❖ This method is not limited to binary or boundary images; you can also apply it directly to a grayscale image to extract corners.
- ❖ The result will provide the coordinates (locations) of all the detected corners.
- ❖ Once you've obtained the corners, you need to decide how to work with them in your application.
- ❖ Harris Corner Detection is probably the most commonly used algorithm for this task.

Summary

- ❖ Representing and describing a segmented region using its boundary is just one of many possible approaches.
- ❖ There are various other methods you can explore, and you should choose the one that best suits your specific application.
- ❖ Once a set of descriptors is extracted, you can save them and use them for tasks such as pattern matching, among other applications.