Assignment 1: Digital Image Processing - Pixel Relationships

Complete Solutions with Detailed Explanations

Subject: Digital Image Processing
Topic: Basic Relationships between Pixels

Q1. Basic Relationships between Pixels (5 Marks)

Discuss the following terminologies related to the Basic Relationships between Pixels with examples:

1. Neighbors of a Pixel
2. Adjacency, Connectivity, Regions, and Boundaries
3. Distance Measures

---

1. Neighbors of a Pixel

Definition A pixel p at coordinates (x, y) has neighbors based on their spatial proximity. The concept of neighbors is fundamental in digital image processing as it defines how pixels relate to each other spatially.

Types of Neighborhoods **4-Neighbors (N4):**

The four horizontal and vertical neighbors of pixel p at (x, y) are:

- (x+1, y) - right neighbor
- (x-1, y) - left neighbor
- (x, y+1) - bottom neighbor
- (x, y-1) - top neighbor

These are the pixels that share an edge with the central pixel.

8-Neighbors (N8):

The 4-neighbors plus the four diagonal neighbors:

- (x+1, y+1) - bottom-right diagonal
- (x-1, y-1) - top-left diagonal
- (x+1, y-1) - top-right diagonal
- (x-1, y+1) - bottom-left diagonal

These include all pixels that share either an edge or a corner with the central pixel.

Diagonal Neighbors (ND):

Only the four diagonal neighbors, which can be expressed as: ND(p) = N8(p) - N4(p)

Visual Example Consider a pixel p at position (2, 2) in a 5×5 image:

```
    Column: 0   1   2   3   4
Row 0:     [ ] [ ] [ ] [ ] [ ]
Row 1:     [ ] [D] [V] [D] [ ]
Row 2:     [ ] [H] [P] [H] [ ]
Row 3:     [ ] [D] [V] [D] [ ]
Row 4:     [ ] [ ] [ ] [ ] [ ]

Legend:
P = Pixel of interest at (2,2)
H = Horizontal neighbors (part of 4-connectivity)
V = Vertical neighbors (part of 4-connectivity)
D = Diagonal neighbors (additional for 8-connectivity)
```

For pixel P at (2,2):

- **N4(p)** = {(2,1), (2,3), (1,2), (3,2)} → 4 neighbors
- **N8(p)** = {(2,1), (2,3), (1,2), (3,2), (1,1), (1,3), (3,1), (3,3)} → 8 neighbors
- **ND(p)** = {(1,1), (1,3), (3,1), (3,3)} → 4 diagonal neighbors only

Boundary Considerations For pixels on the edges or corners of an image:

- Corner pixels have 2 neighbors (4-connectivity) or 3 neighbors (8-connectivity)
- Edge pixels have 3 neighbors (4-connectivity) or 5 neighbors (8-connectivity)
- Interior pixels have 4 neighbors (4-connectivity) or 8 neighbors (8-connectivity)

Practical Applications

- **Edge detection:** Uses neighbor relationships to detect intensity changes
- **Smoothing filters:** Average pixel values with neighbors
- **Morphological operations:** Structuring elements based on neighborhoods
- **Region growing:** Expands regions by examining neighbors

---

2. Adjacency, Connectivity, Regions, and Boundaries

A. Adjacency **Definition:** Two pixels p and q are adjacent if they satisfy two conditions:

1. They are neighbors according to a specified neighborhood type

2. Their intensity values satisfy a specified criterion of similarity (usually their values belong to a predefined set V)

Types of Adjacency:

1. 4-Adjacency: Two pixels p and q with values from set V are 4-adjacent if q is in the set N4(p).

2. 8-Adjacency: Two pixels p and q with values from set V are 8-adjacent if q is in the set N8(p).

3. m-Adjacency (Mixed Adjacency): Two pixels p and q with values from set V are m-adjacent if:

- q is in N4(p), OR
- q is in ND(p) AND the set N4(p)   N4(q) has no pixels whose values are from V

The m-adjacency was introduced to eliminate the ambiguity of multiple paths that can occur with 8-adjacency.

Example of Adjacency:

```
Consider V = {1} (we only consider pixels with value 1)

Image matrix:
  Column: 0  1  2
Row 0:  [ 0  1  0 ]
Row 1:  [ 0  1  1 ]
Row 2:  [ 0  0  1 ]

Let p be the pixel at position (1,1) with value 1.
Let q be the pixel at position (1,2) with value 1.
Let r be the pixel at position (2,2) with value 1.

Analysis:
- p and q are 4-adjacent (q is in N4(p) and both have value 1)
- p and q are also 8-adjacent
- p and q are also m-adjacent
- q and r are 4-adjacent, 8-adjacent, and m-adjacent
- p and r are 8-adjacent (r is diagonal to p)
- p and r are also m-adjacent (checking the m-adjacency rule)
```

B. Connectivity   Definition: A path from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels:

(x , y), (x , y), (x , y), ..., (x , y)

where:

- (x , y) = (x, y) is the starting pixel

- (x , y) = (s, t) is the ending pixel
- Pixels (x , y) and (x , y) are adjacent for 0 i < n

Connected Pixels: Two pixels p and q are said to be connected in a set S if there exists a path from p to q consisting entirely of pixels in S.

Types of Connectivity:

- **4-connected path:** Each successive pixel is 4-adjacent to the previous one
- **8-connected path:** Each successive pixel is 8-adjacent to the previous one
- **m-connected path:** Each successive pixel is m-adjacent to the previous one

Example of Connectivity:

```
Image with V = {1}:
  Column: 0  1  2  3  4
Row 0:  [ 1  1  0  0  1 ]
Row 1:  [ 1  0  0  1  1 ]
Row 2:  [ 0  0  0  0  0 ]
Row 3:  [ 1  1  0  0  0 ]

Using 4-connectivity:
- Path from (0,0) to (1,0): Direct 4-neighbor
- Path from (0,1) to (1,0): (0,1) → (0,0) → (1,0)
- Pixel (0,4) cannot reach (0,0) using only 4-connectivity
- There is no path from (3,0) to (0,0)

Using 8-connectivity:
- Path from (0,0) to (0,4): (0,0) → (0,1) → (1,1) → ... (may not exist in this case)
- The diagonal connections allow more flexible paths
```

C. Regions **Definition:** A region R is a connected subset of pixels. For any two pixels p and q in R, there must exist a path from p to q consisting entirely of pixels in R.

Connected Component: A connected component of a set S is a maximal connected subset of S. This means:

- All pixels in the component are connected to each other
- No pixel outside the component is connected to pixels inside it
- It cannot be enlarged by adding any more adjacent pixels from S

Foreground and Background:

- In binary images, pixels with value 1 typically form the foreground (objects)
- Pixels with value 0 form the background

- Both foreground and background can consist of multiple connected components

Example of Regions:

```
Image with V = {1}:
  Column: 0  1  2  3  4
Row 0:  [ 1  1  0  0  1 ]
Row 1:  [ 1  0  0  1  1 ]
Row 2:  [ 0  0  0  0  0 ]
Row 3:  [ 1  1  0  0  0 ]

Using 4-connectivity:
- Region 1: {(0,0), (0,1), (1,0)} - top-left cluster (3 pixels)
- Region 2: {(0,4), (1,3), (1,4)} - top-right cluster (3 pixels)
- Region 3: {(3,0), (3,1)} - bottom-left cluster (2 pixels)
Total: 3 connected components

Using 8-connectivity:
- Region 1: {(0,0), (0,1), (1,0)} - top-left cluster (3 pixels)
- Region 2: {(0,4), (1,3), (1,4)} - top-right cluster (3 pixels, diagonal at (1,3))
- Region 3: {(3,0), (3,1)} - bottom-left cluster (2 pixels)
Total: 3 connected components

Note: The count is the same in this example, but Region 2 structure differs.
```

D. Boundaries **Definition:** The boundary (also called border, contour, or edge) of a region R is the set of pixels in R that have at least one neighbor that is not in R.

Inner and Outer Boundaries:

- **Inner boundary:** Pixels in R with at least one neighbor outside R
- **Outer boundary:** Pixels outside R with at least one neighbor inside R

Image Boundary: If R is an entire image, then its boundary is defined as the set of pixels in the first and last rows and columns of the image.

Example of Boundaries:

```
Region R (pixels with value 1):
  Column: 0  1  2  3  4
Row 0:  [ 0  0  0  0  0 ]
Row 1:  [ 0  1  1  1  0 ]
Row 2:  [ 0  1  1  1  0 ]
Row 3:  [ 0  1  1  1  0 ]
Row 4:  [ 0  0  0  0  0 ]

Boundary pixels (marked with B):
```

```
  Column: 0  1  2  3  4
Row 0:  [ 0  0  0  0  0 ]
Row 1:  [ 0  B  B  B  0 ]
Row 2:  [ 0  B  1  B  0 ]
Row 3:  [ 0  B  B  B  0 ]
Row 4:  [ 0  0  0  0  0 ]

Analysis:
- The boundary consists of 8 pixels
- Center pixel (2,2) is NOT on the boundary (all neighbors are in R)
- All pixels marked with B have at least one neighbor with value 0
- Using 4-connectivity, the boundary has 8 pixels
- Using 8-connectivity, the boundary would still be 8 pixels in this case
```

Boundary Extraction Algorithm:

The boundary of a region R can be extracted by:

1. Eroding R to obtain R_eroded
2. Boundary = R - R_eroded

Alternatively, for each pixel in R:

- If any of its neighbors is not in R, include it in the boundary

Applications:

- **Object recognition:** Boundaries define object shapes
- **Image segmentation:** Separating regions by their boundaries
- **Contour tracing:** Following boundaries for shape analysis
- **Edge detection:** Finding boundaries between different regions

---

3. Distance Measures

Definition A distance measure (or metric) quantifies the separation between pixels in an image. For pixels p, q, and z with coordinates (x, y), (s, t), and (u, v) respectively, a function D is a distance metric if it satisfies these properties:

1. **Non-negativity:** D(p, q)   0, and D(p, q) = 0 if and only if p = q
2. **Symmetry:** D(p, q) = D(q, p)
3. **Triangle Inequality:** D(p, z)   D(p, q) + D(q, z)

Common Distance Measures

A. Euclidean Distance (D or L norm) Formula:

$$D(p, q) = \sqrt{[(x - s)^2 + (y - t)^2]}$$

Description: This is the straight-line distance between two pixels, representing the true geometric distance.

Example:

```
Given: p = (0, 0) and q = (3, 4)

Calculation:
D (p, q) = √[(0-3)² + (0-4)²]
         = √[(-3)² + (-4)²]
         = √[9 + 16]
         = √25
         = 5 units
```

Geometric Interpretation: The set of pixels at Euclidean distance r from a center pixel (x, y) forms a disk (circle) of radius r.

Pixels at Euclidean distance 2 from center C at (2,2):

```
   0  1  2  3  4
0 [ ][ ][X][ ][ ]   Distance 2.0 from C
1 [ ][X][X][X][ ]   Distance   2.0
2 [X][X][C][X][X]   Distance   2.0
3 [ ][X][X][X][ ]   Distance   2.0
4 [ ][ ][X][ ][ ]   Distance 2.0 from C
```

Applications:

- Medical imaging where true geometric measurements are needed
- Computing actual physical distances
- Circular neighborhood operations
- Pattern matching requiring rotation invariance

B. City-Block Distance (D or L norm or Manhattan Distance) Formula:

```
D (p, q) = |x - s| + |y - t|
```

Description: This measures the distance along horizontal and vertical paths only, like navigating city blocks where you can only move along streets (not diagonally through buildings).

Example:

```
Given: p = (0, 0) and q = (3, 4)

Calculation:
D (p, q) = |0 - 3| + |0 - 4|
         = |-3| + |-4|
         = 3 + 4
         = 7 units
```

Geometric Interpretation: The set of pixels at D distance   r from a center pixel forms a diamond (rotated square) shape.

Pixels at City-Block distance   2 from center C at (2,2):

```
    0  1  2  3  4
0  [ ][ ][X][ ][ ]   Distance 2
1  [ ][X][X][X][ ]   Distance   2
2  [X][X][C][X][X]   Distance   2
3  [ ][X][X][X][ ]   Distance   2
4  [ ][ ][X][ ][ ]   Distance 2
```

Applications:

- Fast computation in algorithms
- 4-connected component analysis
- Grid-based pathfinding
- Operations where diagonal moves are not allowed

C. Chessboard Distance (D or L∞ norm)   Formula:

```
D (p, q) = max(|x - s|, |y - t|)
```

Description: This is the maximum of the horizontal and vertical distances, representing the number of moves a chess king would need to travel from one pixel to another.

Example:

```
Given: p = (0, 0) and q = (3, 4)

Calculation:
D (p, q) = max(|0 - 3|, |0 - 4|)
          = max(|-3|, |-4|)
          = max(3, 4)
          = 4 units
```

Geometric Interpretation: The set of pixels at D distance   r from a center pixel forms a square of side 2r+1.

Pixels at Chessboard distance   2 from center C at (2,2):

```
    0  1  2  3  4
0  [X][X][X][X][X]   Distance 2
1  [X][X][X][X][X]   Distance   2
2  [X][X][C][X][X]   Distance   2
3  [X][X][X][X][X]   Distance   2
4  [X][X][X][X][X]   Distance 2
```

Applications:

- 8-connected component analysis

- Computer graphics and game development
- Operations where diagonal moves are allowed
- Faster approximation of Euclidean distance

Comparison Table

Distance Type	Formula	Example (0,0) to (3,4)	Shape	Connectivity
Euclidean (D)	$\sqrt{[(x\text{-}s)^2 + (y\text{-}t)^2]}$	5.0	Circle	-
City-Block (D)	|x-s| + |y-t|	7	Diamond	4-connected
Chessboard (D)	max(|x-s|, |y-t|)	4	Square	8-connected

Visual Comparison of Distance Measures All pixels at distance 2 from center C:

```
Euclidean (D   2):
      0 1 2 3 4
  0  [ ][ ][X][ ][ ]
  1  [ ][X][X][X][ ]
  2  [X][X][C][X][X]
  3  [ ][X][X][X][ ]
  4  [ ][ ][X][ ][ ]
  (Circular/disk shape)

City-Block (D   2):
      0 1 2 3 4
  0  [ ][ ][X][ ][ ]
  1  [ ][X][X][X][ ]
  2  [X][X][C][X][X]
  3  [ ][X][X][X][ ]
  4  [ ][ ][X][ ][ ]
  (Diamond shape)

Chessboard (D   2):
      0 1 2 3 4
  0  [X][X][X][X][X]
  1  [X][X][X][X][X]
  2  [X][X][C][X][X]
  3  [X][X][X][X][X]
  4  [X][X][X][X][X]
  (Square shape)
```

Relationship Between Distance Measures For any two pixels p and q:

```
D (p, q)   D (p, q)   D (p, q)
```

This means:

- Chessboard distance is always the smallest
- City-block distance is always the largest
- Euclidean distance falls in between

Practical Selection Guidelines **Use Euclidean distance when:**

- True geometric measurements are required
- Working with medical or scientific imaging
- Rotation invariance is needed
- Physical accuracy is paramount

Use City-Block distance when:

- Computational efficiency is important
- Working with 4-connected operations
- Grid-based pathfinding (like in some games or robots)
- Diagonal movement is restricted

Use Chessboard distance when:

- Working with 8-connected operations
- Fast approximation is acceptable
- Diagonal movement is allowed
- Processing speed is critical

Q2. Connectivity Analysis (5 Marks)

Problem Statement

Explain the 4, 8, and m connectivity of pixels. Consider the two image subsets S and S shown below. For V = {1}, determine how many:

- (a) 4-connected components
- (b) 8-connected components
- (c) m-connected components

are there in S and S ? Are S and S adjacent?

Given Image Subsets

```
       S                     S
    0  1  2  3  4        5  6  7  8  9
0 [ 0  0  0  0  0 ]    [ 0  0  1  1  0 ]
1 [ 0  0  1  0  0 ]    [ 1  0  0  0  0 ]
2 [ 0  0  1  0  1 ]    [ 1  0  0  0  0 ]
3 [ 0  1  1  1  0 ]    [ 0  1  1  1  1 ]
4 [ 0  1  1  1  0 ]    [ 0  1  1  1  1 ]
```

Given: V = {1} (we only consider pixels with value 1)

Detailed Explanation of Connectivity Types

1. Four-Connectivity (4-Connectivity) **Definition:** Two pixels p and q are 4-connected if:

1. Both pixels have values from the set V
2. q is in N (p), meaning q is a horizontal or vertical neighbor of p

Neighbor Set: For pixel p at (x, y), the 4-neighbors are:

- N (p) = {(x±1, y), (x, y±1)}

Path: A 4-connected path between pixels p and q consists of pixels where each successive pixel in the path is a 4-neighbor of the previous one.

Connected Component: A 4-connected component is a maximal set of pixels where every pixel can be reached from every other pixel via a 4-connected path.

Visual Example:

```
These pixels are 4-connected:
  0  1  2
0 [ ][1][ ]
1 [1][1][1]
2 [ ][1][ ]

All five 1-valued pixels form one 4-connected component.
```

2. Eight-Connectivity (8-Connectivity) **Definition:** Two pixels p and q are 8-connected if:

1. Both pixels have values from the set V
2. q is in N (p), meaning q is a horizontal, vertical, or diagonal neighbor of p

Neighbor Set: For pixel p at (x, y), the 8-neighbors are:

- N (p) = {(x±1, y), (x, y±1), (x±1, y±1)}

Path: An 8-connected path allows movement in all 8 directions (horizontal, vertical, and diagonal).

Connected Component: An 8-connected component is a maximal set of pixels where every pixel can be reached from every other pixel via an 8-connected path.

Visual Example:

```
These pixels are 8-connected:
  0  1  2
0 [1][ ][1]
```

```
1 [ ][1][ ]
2 [1][ ][1]

All five 1-valued pixels form one 8-connected component
(connected through diagonal neighbors).
```

Problem with 8-Connectivity: 8-connectivity can lead to ambiguous situations where multiple paths exist between the same two pixels, which can cause issues in some image processing algorithms.

3. Mixed Connectivity (m-Connectivity) **Definition:** Two pixels p and q are m-connected if:

1. Both pixels have values from the set V
2. Either:
   - q is in N (p) (4-neighbor), OR
   - q is in N (p) (diagonal neighbor) AND the intersection N (p)   N (q) contains NO pixels with values from V

Purpose: m-connectivity was introduced to eliminate the multiple path ambiguity that can occur with 8-connectivity while still allowing diagonal connections where appropriate.

Rule Explanation:

- Always allow 4-neighbor connections (horizontal and vertical)
- Allow diagonal connections ONLY when there is no 4-connected path through their common 4-neighbors

Visual Example:

```
Configuration 1:
  0  1  2
0 [ ][1][ ]
1 [1][0][1]
2 [ ][ ][ ]

Pixels at (0,1) and (1,2) are:
- NOT 4-connected
- 8-connected (diagonal)
- m-connected (their common 4-neighbor (1,1) has value 0)

Configuration 2:
  0  1  2
0 [ ][1][ ]
1 [1][1][1]
2 [ ][ ][ ]

Pixels at (0,1) and (1,2) are:
```

```
- NOT 4-connected
- 8-connected (diagonal)
- NOT m-connected (their common 4-neighbor (1,1) has value 1)
```

Solution Analysis

Pixel Identification **Pixels with value 1 in S :**

- Row 1: (1,2)
- Row 2: (2,2), (2,4)
- Row 3: (3,1), (3,2), (3,3)
- Row 4: (4,1), (4,2), (4,3)

Pixels with value 1 in S :

- Row 0: (0,7), (0,8)
- Row 1: (1,5)
- Row 2: (2,5)
- Row 3: (3,6), (3,7), (3,8), (3,9)
- Row 4: (4,6), (4,7), (4,8), (4,9)

(a) 4-Connected Components

Analysis for S **Step 1: Identify all pixels with value 1**

```
S visualization (only showing 1s):
    0  1  2  3  4
0  [ ][ ][ ][ ][ ]
1  [ ][ ][1][ ][ ]
2  [ ][ ][1][ ][1]
3  [ ][1][1][1][ ]
4  [ ][1][1][1][ ]
```

Step 2: Check 4-connectivity (vertical and horizontal only)

Starting from (1,2):

- (1,2) → (2,2) [vertical neighbor, connected]

From (2,2):

- (2,2) → (3,2) [vertical neighbor, connected]

From (3,2):

- (3,2) → (3,1) [horizontal neighbor, connected]
- (3,2) → (3,3) [horizontal neighbor, connected]
- (3,2) → (4,2) [vertical neighbor, connected]

From (4,2):

- (4,2) → (4,1) [horizontal neighbor, connected]
- (4,2) → (4,3) [horizontal neighbor, connected]

From (3,1):

- (3,1) → (4,1) [vertical neighbor, connected]

From (3,3):

- (3,3) → (4,3) [vertical neighbor, connected]

Component 1: {(1,2), (2,2), (3,1), (3,2), (3,3), (4,1), (4,2), (4,3)}

Now checking (2,4):

- (2,4) has no 4-neighbors with value 1
- (2,4) is isolated

Component 2: {(2,4)}

Result for S : 2 components (4-connected)

Analysis for S Step 1: Identify all pixels with value 1

```
S  visualization (only showing 1s):
    5  6  7  8  9
0 [ ][ ][1][1][ ]
1 [1][ ][ ][ ][ ]
2 [1][ ][ ][ ][ ]
3 [ ][1][1][1][1]
4 [ ][1][1][1][1]
```

Step 2: Check 4-connectivity

Group 1: Top row

- (0,7) → (0,8) [horizontal neighbor, connected]

Component 1: {(0,7), (0,8)}

Group 2: Left column

- (1,5) → (2,5) [vertical neighbor, connected]

Component 2: {(1,5), (2,5)}

Group 3: Bottom block

- (3,6) → (3,7) [horizontal neighbor, connected]
- (3,7) → (3,8) [horizontal neighbor, connected]
- (3,8) → (3,9) [horizontal neighbor, connected]
- (3,6) → (4,6) [vertical neighbor, connected]
- (3,7) → (4,7) [vertical neighbor, connected]

- (3,8) → (4,8) [vertical neighbor, connected]
- (3,9) → (4,9) [vertical neighbor, connected]
- (4,6) → (4,7) [horizontal neighbor, connected]
- (4,7) → (4,8) [horizontal neighbor, connected]
- (4,8) → (4,9) [horizontal neighbor, connected]

Component 3: {(3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

Result for S : 3 components (4-connected)

(b) 8-Connected Components

Analysis for S Additional connections with diagonal neighbors:

Previous 4-connected component remains: {(1,2), (2,2), (3,1), (3,2), (3,3), (4,1), (4,2), (4,3)}

Now checking (2,4) for diagonal connections:

- (2,4) and (3,3) are diagonal neighbors
- (2,4) → (3,3) [diagonal neighbor, connected]

All pixels now form one component: {(1,2), (2,2), (2,4), (3,1), (3,2), (3,3), (4,1), (4,2), (4,3)}

Result for S : 1 component (8-connected)

Analysis for S Checking for additional diagonal connections:

Component 1: {(0,7), (0,8)} - unchanged

Component 2: {(1,5), (2,5)}

- Check if (2,5) connects to (3,6): YES, diagonal neighbors

Component 3: {(3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

Since (2,5) and (3,6) are diagonal neighbors: **New Component 2:** {(1,5), (2,5), (3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

Result for S : 2 components (8-connected)

- Component 1: {(0,7), (0,8)}
- Component 2: {(1,5), (2,5), (3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

(c) m-Connected Components

Analysis for S Applying m-connectivity rules:

Start with all 4-connected relationships (same as part a).

Now check diagonal connections:

Checking (2,2) and (3,1):

- They are diagonal neighbors
- N (2,2) = {(1,2), (3,2), (2,1), (2,3)}
- N (3,1) = {(2,1), (4,1), (3,0), (3,2)}
- N (2,2)   N (3,1) = {(2,1), (3,2)}
- (3,2) has value 1 (belongs to V)
- Therefore, NOT m-connected via diagonal

Checking (2,2) and (3,3):

- They are diagonal neighbors
- N (2,2) = {(1,2), (3,2), (2,1), (2,3)}
- N (3,3) = {(2,3), (4,3), (3,2), (3,4)}
- N (2,2)   N (3,3) = {(2,3), (3,2)}
- (3,2) has value 1 (belongs to V)
- Therefore, NOT m-connected via diagonal

Checking (2,4) and (3,3):

- They are diagonal neighbors
- N (2,4) = {(1,4), (3,4), (2,3), (2,5)}
- N (3,3) = {(2,3), (4,3), (3,2), (3,4)}
- N (2,4)   N (3,3) = {(2,3), (3,4)}
- (2,3) has value 0
- (3,4) has value 0
- Neither belongs to V
- Therefore, m-connected via diagonal

Result: The main 4-connected component connects to (2,4) via m-connectivity.

Result for S : 1 component (m-connected) {(1,2), (2,2), (2,4), (3,1), (3,2), (3,3), (4,1), (4,2), (4,3)}

Analysis for S   Applying m-connectivity rules:

Start with 4-connected components:

- Component 1: {(0,7), (0,8)}
- Component 2: {(1,5), (2,5)}
- Component 3: {(3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

Checking (2,5) and (3,6):

- They are diagonal neighbors
- N (2,5) = {(1,5), (3,5), (2,4), (2,6)}
- N (3,6) = {(2,6), (4,6), (3,5), (3,7)}
- N (2,5)   N (3,6) = {(2,6), (3,5)}
- (2,6) has value 0

- (3,5) has value 0
- Neither belongs to V
- Therefore, m-connected via diagonal

Result: Components 2 and 3 merge.

Result for S : 2 components (m-connected)

- Component 1: {(0,7), (0,8)}
- Component 2: {(1,5), (2,5), (3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)}

Adjacency of S and S

Definition: Two regions S and S are adjacent if at least one pixel from S (with value in V) is a neighbor of at least one pixel from S (with value in V).

Analysis:

Boundary between S and S :

- S occupies columns 0-4
- S occupies columns 5-9
- They are adjacent at the boundary between columns 4 and 5

Checking 4-adjacency:

Rightmost 1-valued pixels in S :

- (2,4) at column 4

Leftmost 1-valued pixels in S :

- (1,5) at column 5
- (2,5) at column 5

Checking pixel pairs:

- (2,4) and (2,5):
  - Row: both at row 2
  - Columns: 4 and 5 (adjacent columns)
  - They are horizontal neighbors (4-adjacent)
  - Both have value 1 (belong to V)

Conclusion: Since (2,4) from S and (2,5) from S are 4-adjacent and both have values in V = {1}, the regions S and S are adjacent.

Answer: YES, S and S are adjacent.

Summary of Results

Connected Components Count

Connectivity Type	S Components	S Components
4-connected	2	3
8-connected	1	2
m-connected	1	2

Detailed Component Breakdown

S Components:

4-connectivity:

1. Component 1: {(1,2), (2,2), (3,1), (3,2), (3,3), (4,1), (4,2), (4,3)} - 8 pixels
2. Component 2: {(2,4)} - 1 pixel

8-connectivity:

1. Component 1: All pixels - 9 pixels total

m-connectivity:

1. Component 1: All pixels - 9 pixels total

S Components:

4-connectivity:

1. Component 1: {(0,7), (0,8)} - 2 pixels
2. Component 2: {(1,5), (2,5)} - 2 pixels
3. Component 3: {(3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)} - 8 pixels

8-connectivity:

1. Component 1: {(0,7), (0,8)} - 2 pixels
2. Component 2: {(1,5), (2,5), (3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)} - 10 pixels

m-connectivity:

1. Component 1: {(0,7), (0,8)} - 2 pixels
2. Component 2: {(1,5), (2,5), (3,6), (3,7), (3,8), (3,9), (4,6), (4,7), (4,8), (4,9)} - 10 pixels

Adjacency Result

Are S and S Adjacent? Answer: YES

Reason: Pixels (2,4) from S and (2,5) from S are 4-adjacent (horizontal neighbors) and both have value 1, which belongs to the set V = {1}.

Key Takeaways

Understanding Connectivity

1. **4-connectivity** is the most restrictive, allowing only horizontal and vertical connections
2. **8-connectivity** is the most permissive, allowing diagonal connections freely
3. **m-connectivity** provides a middle ground, using diagonal connections only when necessary to avoid path ambiguity

Practical Implications

When to use each connectivity type:

- **4-connectivity:**
  - When diagonal connections should not be allowed
  - Grid-based applications (like some pathfinding algorithms)
  - When analyzing images with rectangular structures
- **8-connectivity:**
  - When diagonal connections are natural and expected
  - Most general-purpose image processing tasks
  - When processing images with arbitrary orientations
- **m-connectivity:**
  - When you need to avoid path ambiguities
  - Advanced image segmentation algorithms
  - When precise connectivity definition is critical

Common Applications

Connected component labeling: Used in:

- Object counting
- Blob analysis
- Optical character recognition (OCR)
- Medical image analysis

Region growing: Used in:

- Image segmentation
- Flood fill algorithms
- Image editing tools

Boundary following: Used in:

- Shape analysis
- Object recognition
- Contour extraction