***Question1***

**1. Neighbors**

In image processing, the neighbors of a pixel are the pixels that are directly adjacent to it. The two most common types of neighbors are:

* **4-neighbors (4-connectivity)**: The pixels that are directly horizontal and vertical neighbors.
* **8-neighbors (8-connectivity)**: The pixels that are horizontal, vertical, and diagonal neighbors.

**Diagram**:

mathematica

Copy code

4-neighbors:

N

W P E

S

8-neighbors:

NW N NE

W P E

SW S SE

Where P is the pixel of interest, N is the north neighbor, S is the south neighbor, E is the east neighbor, W is the west neighbor, and the diagonal neighbors are labeled accordingly.

**2. Connectivity of Pixels**

Connectivity defines how pixels are connected to each other, typically through their neighbors. The two primary types of connectivity are:

* **4-connectivity**: A pixel is connected to its 4-neighbors (N, S, E, W).
* **8-connectivity**: A pixel is connected to its 8-neighbors (including diagonal neighbors).

**Diagram**:

yaml

Copy code

4-connectivity:

|

--P--

|

8-connectivity:

\ | /

-- P --

/ | \

In 4-connectivity, the connections are directly horizontal and vertical. In 8-connectivity, the connections include diagonal neighbors as well.

**3. Euclidean Distance**

Euclidean distance is the straight-line distance between two pixels in an image. For two pixels (x1,y1)(x\_1, y\_1)(x1​,y1​) and (x2,y2)(x\_2, y\_2)(x2​,y2​), the Euclidean distance ddd is calculated as: d=(x2−x1)2+(y2−y1)2d = \sqrt{(x\_2 - x\_1)^2 + (y\_2 - y\_1)^2}d=(x2​−x1​)2+(y2​−y1​)2​

**Diagram**:

scss

Copy code

P1(x1, y1) ---------d---------

\

\

\

P2(x2, y2)

The line connecting P1P1P1 and P2P2P2 represents the Euclidean distance.

**4. City Block Distance (Manhattan Distance)**

City block distance, also known as Manhattan distance, is the distance between two points measured along axes at right angles. For two pixels (x1,y1)(x\_1, y\_1)(x1​,y1​) and (x2,y2)(x\_2, y\_2)(x2​,y2​), the Manhattan distance ddd is: d=∣x2−x1∣+∣y2−y1∣d = |x\_2 - x\_1| + |y\_2 - y\_1|d=∣x2​−x1​∣+∣y2​−y1​∣

**Diagram**:

scss

Copy code

P1(x1, y1) ---------d---------

|

|

|

P2(x2, y2)

The movement resembles a grid-like path, akin to navigating city blocks.

**5. Chess Board Distance**

Chess board distance, also known as Chebyshev distance, is the maximum of the absolute differences between the coordinates of two points. For two pixels (x1,y1)(x\_1, y\_1)(x1​,y1​) and (x2,y2)(x\_2, y\_2)(x2​,y2​), the Chess board distance ddd is: d=max⁡(∣x2−x1∣,∣y2−y1∣)d = \max(|x\_2 - x\_1|, |y\_2 - y\_1|)d=max(∣x2​−x1​∣,∣y2​−y1​∣)

**Diagram**:

scss

Copy code

P1(x1, y1) ---------d---------

\ /

\ /

\ /

P2(x2, y2)

This distance measures the number of moves a king would take on a chess board to move from one point to another, allowing for both horizontal/vertical and diagonal moves.

These concepts are fundamental in image processing and computer vision, providing the basis for operations such as segmentation, filtering, and object detection.

***Question2***

### 1. Derivative Filters

**Example/Technique**: **Sobel Filter**

Derivative filters are used to detect edges in images by calculating the gradient of image intensity at each pixel.

**Diagram**:

less

Copy code

Original Image:

[ 10 10 10 10 10 ]

[ 10 50 50 50 10 ]

[ 10 50 100 50 10 ]

[ 10 50 50 50 10 ]

[ 10 10 10 10 10 ]

Sobel Filter (X-direction):

[ -1 0 1 ]

[ -2 0 2 ]

[ -1 0 1 ]

Filtered Image:

[ 0 0 0 0 0 ]

[ 0 80 160 80 0 ]

[ 0 100 200 100 0 ]

[ 0 80 160 80 0 ]

[ 0 0 0 0 0 ]

### 2. Template Matching Filters

**Example/Technique**: **Normalized Cross-Correlation**

Template matching filters are used to find parts of an image that match a template image.

**Diagram**:

less

Copy code

Original Image: Template:

[ 10 10 10 10 10 ] [ 50 50 ]

[ 10 50 50 50 10 ] [ 50 50 ]

[ 10 50 100 50 10 ]

[ 10 50 50 50 10 ]

[ 10 10 10 10 10 ]

Matching Result:

[ 0 0 0 0 0 ]

[ 0 1 1 1 0 ]

[ 0 1 1 1 0 ]

[ 0 1 1 1 0 ]

[ 0 0 0 0 0 ]

Here, '1' indicates a strong match of the template.

### 3. Gaussian Filters

**Example/Technique**: **Gaussian Blur**

Gaussian filters are used to blur images and remove noise. They work by convolving the image with a Gaussian function.

**Diagram**:

less

Copy code

Original Image:

[ 10 10 10 10 10 ]

[ 10 50 50 50 10 ]

[ 10 50 100 50 10 ]

[ 10 50 50 50 10 ]

[ 10 10 10 10 10 ]

Gaussian Kernel:

[ 1 4 7 4 1 ]

[ 4 16 26 16 4 ]

[ 7 26 41 26 7 ]

[ 4 16 26 16 4 ]

[ 1 4 7 4 1 ]

Blurred Image:

[ 24 30 33 30 24 ]

[ 30 45 54 45 30 ]

[ 33 54 76 54 33 ]

[ 30 45 54 45 30 ]

[ 24 30 33 30 24 ]

### 4. Pattern Fit Algorithms

**Example/Technique**: **Least Squares Fitting**

Pattern fit algorithms find a curve or function that best fits a set of data points. In image processing, this can be used for edge fitting or shape recognition.

**Diagram**:

less

Copy code

Data Points: Fitted Curve:

[ (1, 2) (2, 4) ] y = 2x

[ (3, 6) (4, 8) ]

[ (5, 10) (6, 12) ]

### 5. Hough Transforms

**Example/Technique**: **Line Detection**

Hough Transform is used to detect lines, circles, or other shapes in an image.

**Diagram**:

less

Copy code

Original Image: Hough Space:

[ 0 0 0 0 0 ] [ 1 1 1 1 1 ]

[ 0 1 1 1 0 ] [ 0 0 1 0 0 ]

[ 0 1 0 1 0 ] [ 0 1 0 1 0 ]

[ 0 1 1 1 0 ] [ 0 0 0 0 0 ]

[ 0 0 0 0 0 ] [ 0 0 0 0 0 ]

Detected Line: Hough Space (Transformed):

[ 1 0 0 0 1 ] [ r = x cos(θ) + y sin(θ) ]

[ 0 1 0 1 0 ]

[ 0 0 1 0 0 ]

[ 0 1 0 1 0 ]

[ 1 0 0 0 1 ]

Each technique and diagram above provides a clear illustration of the concept and its application in image processing.