CSE438: Image Processing Lab Manual

Lab-01: Image Processing Operations

Objective:
To understand and implement fundamental image processing techniques such as perimeter determination, thresholding, connectivity, distance measurement, arithmetic operations, logical operations, contrast adjustment, brightness enhancement, quantization, and digital negation.

1. Determine the Perimeter of an Object using 4-connected and 8-connected Neighborhoods

Theory: The perimeter of an object in a binary image can be determined by analyzing its boundary pixels. The two primary methods are:

- **4-connected neighborhood:** A pixel is considered connected if it shares an edge with another pixel.
- **8-connected neighborhood:** A pixel is considered connected if it shares either an edge or a corner with another pixel.

Implementation Steps:

1. Convert the input image into a binary format.
2. Identify boundary pixels using 4-connected and 8-connected connectivity.
3. Compute and compare the perimeters.
4. Use bwperim() function to find out the perimeter.

2. Create a Binary Image Using a Threshold

Theory: Thresholding is a segmentation technique where pixels are classified as foreground or background based on a threshold value.

Implementation Steps:

1. Convert the grayscale image to binary using a threshold (e.g., Otsu's method or a fixed threshold).
2. Display the binary image.

3. Determine the Number of Objects in a Binary Image Using Connectivity

Theory: Objects in a binary image can be counted using connected component labeling (CCL), which assigns unique labels to different objects based on their connectivity.

Implementation Steps:

1. Perform connected component labeling using 4-connected or 8-connected components.

2. Count the number of unique labeled regions.

4. Find the Euclidean Distance Between Two Points in an Image

Theory: The Euclidean distance between two points (x1,y1)(x_1, y_1) and (x2,y2)(x_2, y_2) is given by: d=(x2−x1)2+(y2−y1)2d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}

Implementation Steps:

1. Select two points manually or programmatically.
2. Compute the Euclidean distance.
3. Display the computed distance.

5. Perform Arithmetic Operations on Images

Operations:

- **Addition**: Adds pixel values of two images.
- **Subtraction**: Subtracts pixel values of one image from another.
- **Multiplication**: Multiplies pixel values of two images.
- **Division**: Divides pixel values of one image by another.

Implementation Steps:

1. Convert images to the same size and type.
2. Apply arithmetic operations.
3. Display results.

6. Perform Logical Operations on Images

Operations:

- **AND**: Logical AND between pixel values.
- **OR**: Logical OR between pixel values
- **NOT**: Logical negation of pixel values.

Implementation Steps:

1. Convert images to binary if necessary.
2. Apply logical operations.

3. Display results.

7. Adjust the Contrast of an Image

Theory: Contrast adjustment enhances the differences between light and dark areas of an image.

Implementation Steps:

1. Normalize pixel values.
2. Apply contrast stretching or histogram equalization.
3. Display the enhanced image.

8. Brighten an Image

Theory: Brightness adjustment increases the intensity of pixel values.

Implementation Steps:

1. Increase pixel values by a fixed amount.
2. Ensure pixel values remain within valid range (0-255).
3. Display the brightened image.

9. Quantize the Grayscale Image by 8 Levels

Theory: Quantization reduces the number of intensity levels in an image, leading to a simpler representation.

Implementation Steps:

1. Divide pixel values into 8 discrete levels.
2. Assign each pixel to the closest level.
3. Display the quantized image.

10. Find the Digital Negative of an Image

Theory: Digital negative transformation inverts pixel values: IMG=255−img

Implementation Steps:

1. Subtract pixel values from 255.
2. Display the negative image.

Functions:

1. **bwperim(binaryImage/image, connectivity)**: Detects the perimeter (boundary) of objects in a binary image.

Parameters:

binaryImage → A binary image where objects are represented by 1s (white) and background by 0s (black).

connectivity → Can be 4 (horizontal/vertical neighbors) or 8 (includes diagonal neighbors).

2. **imbinarize(image, threshold):**

Purpose: Converts a grayscale image to a binary image based on a threshold.

Parameters:

image → Grayscale image.

threshold → Intensity level (between 0 and 1) for binarization.

3. **bwconncomp(binaryImage, connectivity):**

Purpose: Finds connected components (objects) in a binary image.

Parameters:

binaryImage → Binary image where objects are white (1).

connectivity → 4 or 8 (similar to bwperim).

4. **norm([x2-x1, y2-y1]):**

Purpose: Computes the Euclidean distance (shortest straight-line distance) between two points.

Formula:

Parameters:

(x1, y1) and (x2, y2) → Coordinates of two points.

5. **imadd(A, B), imsubtract(A, B), immultiply(A, B), imdivide(A, B):**

Purpose: Performs pixel-wise operations between two images.

6.

bitand(A, B) → Performs AND operation.

bitor(A, B) → Performs OR operation.

bitcmp(A) → Performs NOT operation.

7.

floor(image / (256 / levels)) * (256 / levels) → Reduces intensity levels.

levels=numoflevels