cis111-2025-1-e4Project2: Edge Detection for Black & White Images

Common Material

Consider `cis000_common` package.

1. Update `StudentInfo` with your info.
2. Read `.pdf` files in `cis000_common`.

Introduction

Edge detection is a fundamental technique in **image processing** and **computer vision**, used to identify boundaries between different regions in an image. These edges often represent significant changes in pixel intensity, which can be useful in applications like **object detection, medical imaging, and autonomous vehicles**.

In this project, you will implement the **Laplace Edge Detection** algorithm using a **4-kernel** approach specifically designed for **black & white (binary) images** (where pixels are either 0=black or 255=white).

Problem Definition

Given a **2D black & white image** (a binary matrix where pixels are either `0` or `255`), your task is to:

1. Apply the **4-kernel Laplace operator** to detect edges.
2. Handle **image borders** (using zero-padding or mirroring).
3. **Binarize the output** (convert edge pixels to `255` and non-edges to `0`).
4. **Display/print** the edge-detected image.

Input:

- A 2D binary array (`int[][] image`) where:
  - `0` = Black

    - `255` = White
- Image dimensions (`width`, `height`).

Output:

- A new 2D binary array (`int[][] edges`) where edges are `255` and non-edges are `0`.

Laplace 4-Kernel for Binary Images

The **Laplace 4-kernel** detects edges by comparing a pixel with its **four neighbors (top, bottom, left, right)**.

Kernel Structure:

0	**-1**	**0**
-1	**4**	**-1**
0	**-1**	**0**

How It Works on Binary Images:

- If a pixel is **surrounded by the same color**, the Laplace response is `0` (no edge).
- If a pixel **differs from its neighbors**, the response is non-zero (edge detected).

Mathematical Formulation:

For a binary pixel at `(x, y)`:

$$L(x, y) = 4 \cdot I(x, y) - I(x-1, y) - I(x+1, y) - I(x, y-1) - I(x, y+1)$$

- If `L(x,y) != 0`, the pixel is an edge (`255`).
- Else, it's not an edge (`0`).

Implementation Steps of `edgeDetector` Method

Step 1: Define the Laplace Kernel

```
final int[][] LAPLACE_KERNEL = {
 {0, -1, 0},
 {-1, 4, -1},
 {0, -1, 0}
};
```

This step is complete; proceed with the remaining tasks.

Step 2: Apply Convolution & Binarize

- Loop through each pixel (excluding borders).
- Compute the Laplace response.
- If response ≠ **0**, set output to `255` (edge), else `0`.

Step 3: Handle Borders

- **Zero-padding:** Treat out-of-bound pixels as `0` (black).
- **Mirroring:** Reflect edge pixels.

Step 4: Display Result

Print the edge-detected binary matrix.

Implementation Steps of `edgePlot` Method

After obtaining the edge-detected binary matrix, represent it visually using `.` for edges (255) and ` ` spaces for non-edges (0). Store this ASCII representation in a String and return it from the method.