# Hole Filling Project

## Overview

This project implements image processing library that fills holes in images, along with a small command line utility that uses that library. It provides a library for filling holes based on weighted interpolation algorithms and a command-line utility for processing images with holes. The implementation supports both the default hole-filling algorithm and an approximate method for faster execution.

## Features

- Supports grayscale image hole filling.

- Uses weighted interpolation to estimate missing pixel values.

- Supports both 4-connected and 8-connected pixel connectivity.

- Configurable weight function with parameters z and epsilon.

- Implements an approximate hole-filling algorithm for improved performance.

- Command-line utility for easy execution.

## Algorithm Explanation

The process follows these steps:

1. **Image Preprocessing**:

  - Converts the image to grayscale.

  - Identifies hole pixels (-1 values) and boundary pixels.

  - Supports 4-connected and 8-connected pixel connectivity.

2. **Weight Function**:

  - Configurable z and epsilon parameters.

3. **Hole Filling**:

  - Computes new values for missing pixels based on boundary pixel contributions.

  - Uses either the default or approximate filling algorithm.

## Complexity Analysis

- **Default Algorithm**: O(nm), where n is the number of hole pixels and m is the number of boundary pixels.

- **Approximate Algorithm**:

Faster alternative but less accurate, at O(n) time complexity.

## Usage

Run the command-line utility with:

java -cp bin CommandLineUtility.Main <image-path> <mask-path> <connectivity> <z> <epsilon>

Example:

java -cp bin CommandLineUtility.Main image.png mask.png 8 3 0.01

- **image-path**: Path to the input image.

- **mask-path**: Path to the mask defining the hole.

- **connectivity**: 4 or 8 for pixel connectivity type.

- **z**: Weight function exponent.

- **epsilon**: Small constant to avoid division by zero.

## File Structure

```

|-- src/

|  |-- CommandLineUtility/

|  |  |-- Main.java      # Entry point for CLI execution

|  |-- lib/

|  |  |-- ImagePreProcessing.java  # Preprocessing operations

|  |  |-- ImageLibraryManager.java # Image handling

|  |  |-- entities/

|  |  |  |-- Pixel.java          # Pixel representation

|  |  |  |-- ProcessedImageFields.java # Image processing data representation

|  |  |-- algorithms/

|  |  |  |-- AlgorithmManager.java      # Algorithm Strategy.

|  |  |  |-- FillingAlgorithm.java      # Abstract class for the algorithms.
```

| | | |-- DefaultHoleFillingAlgorithm.java # Default algorithm

| | | |-- ApproximateAlgorithm.java  # Approximate algorithm

| | | |-- WeightFunction.java  # Interface for weight functions

|-- README.md

Example Results

- **Input Image:** Lenna.png

- **Mask:** Mask.png

- **Output:** Lenna_FILLED.png

**Usage Example**

- Create an ImageLibraryManager Object for using the library functions and classes.
- Use the processor for processing the images and creating the grayscale pixel array, and retrieve the ProcessedImageFields output, which is an object made of the Pixel array, HashSet of the Boundary pixels and HashSet of the Hole pixels.
- Use the AlgorithmManager to manage and choose the Algorithm or Weight Function
- Use the pixel Array Output to save the image or do something else.