**HCMUS - VNUHCM / FIT /**

**Computer Vision & Cognitive Cybernetics Department**

**Digital Image and Video Processing Application**

**Student ID:** 21127690

**Student name:** Ngo Nguyen Thanh Thanh

**Report: Morphological Operations** **for Grayscale Image**

**I. Evaluation summary:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Types** | **Task** | **Implemetation (Without OpenCV)** | **Implementation (With OpenCV)** | **Completion (%)** |
| **1** | **Gray scale** | **Grayscale Dilation** | Implemented using max filter over neighborhood | Uses cv2.dilate() | 100% |
| **2** | **Grayscale Erosion** | Implemented using min filter over neighborhood | Uses cv2.erode() | 100% |
| **3** | **Grayscale Opening** | Implemented using grayscale erosion followed by dilation | Uses cv2.morphologyEx(…, cv2.MORPH\_OPEN) | 100% |
| **4** | **Grayscale Closing** | Implemented using grayscale dilation followed by erosion | Uses cv2.morphologyEx(…, cv2.MORPH\_CLOSE) | 100% |
| **5** | **Grayscale smoothing** | Implemented using Gaussian or median filtering |  | 100% |
| **6** | **Grayscale Morphology Gradient** | Implemented using difference between dilation and erosion | Uses cv2.morphologyEx(…, cv2.MORPH\_GRADIENT) | 100% |
| **7** | **Top-hat transformation** | Implemented using difference between original image and opened image | Uses cv2.morphologyEx(…, cv2.MORPH\_TOPHAT) | 100% |
| **8** | **Textural segmentation** | Implemented using morphological filtering and thresholding |  | 100% |
| **9** | **Granulometry** | Implemented using morphological opening with increasing kernel sizes |  | 100% |
| **10** | **Reconstruction** | Implemented using marker-controlled dilation until convergence |  | 100% |

1. **List of features and file structure:**

**File structure:**

**A screenshot of a computer

Description automatically generated**

**input\_image/** – Stores input images.

**output\_image/** – Contains processed output images.

**source/ – Main source code directory.**

* **morphological\_operations/** – Contains morphological processing modules.
  + **grayscale.py** – Handles grayscale image operations.
* **main\_grayscale.py** – Executes grayscale image processing.

**Functions and methods used:**

**1**. **grayscale.py**

**Main functions:**  
**pad\_image\_grayscale(img, kernel, pad\_value)** – Adds constant padding (value = 0 for dilation, 255 for erosion) to preserve output size and prevent border errors during processing.

• **grayscale\_dilation(img, kernel**) – Applies grayscale dilation (f ⊕ b) to brighten dark regions by computing the local maximum of (image + kernel).

**• grayscale\_erosion(img, kernel)** – Applies grayscale erosion (f ⊖ b) to darken bright regions by computing the local minimum of (image - kernel).

• **grayscale\_opening(img, kernel)** – Performs grayscale opening (f ∘ b) by erosion followed by dilation; removes small bright noise and smooths object boundaries.

**• grayscale\_closing(img, kernel)** – Performs grayscale closing (f • b) by dilation followed by erosion; fills small dark gaps and connects adjacent objects.

**• grayscale\_smoothing(img, kernel)** – Smooths image by applying opening then closing; suppresses both small bright and dark noise.

**• grayscale\_morphology\_gradient(img, kernel)** – Computes morphological gradient: highlights object boundaries by calculating (dilation - erosion).

**• top\_hat(img, kernel)** – Extracts small bright features using Top-Hat transformation: (original - opening).

**• textural\_segmentation(img, kernel)** – Performs texture-based segmentation using Top-Hat to highlight fine bright texture elements.

**• granulometry(img, sizes)** – Analyzes object size distribution by applying opening with multiple structuring element sizes and summing pixel intensities.

**• reconstruction(marker, mask, kernel, max\_iter=100)** – Performs morphological reconstruction by iterative dilation of marker under mask constraint; useful for restoring image structures.

**• create\_structuring\_element(size)** – Creates a square structuring element (kernel) of a specified size.

**2. main\_grayscale.py**

**Main Functions:**

* Load and preprocess the input image as a grayscale image from the specified file path.
* Provide two processing modes for morphological operations:
  + Manual: Implements grayscale morphological operations using custom algorithms defined in grayscale.py.
  + OpenCV: Uses built-in OpenCV functions for standard morphological processing.
* Support morphological operations:  
  Original, Dilate, Erode, Open, Close, Smoothing, Gradient, TopHat, TexturalSegmentation, Granulometry, Reconstruction.
* Display and save output images using OpenCV and write granulometry results to text/plot files.
* Measure and report execution time for performance comparison between manual and OpenCV implementations.

**Function Descriptions:**

* **apply\_manual(img, kernel, seed=(10,10))**  
  Applies custom grayscale morphological operations using implementations from grayscale.py. Returns a dictionary containing operation names and their corresponding processed images. Also includes granulometry values (as a list) and image reconstruction results.
* **apply\_opencv(img, kernel, seed=(10,10))**Applies standard grayscale morphological operations using OpenCV. Returns a dictionary of images corresponding to each operation.
* **operator(in\_file, out\_file, mor\_op, mode, wait\_key\_time=5000)**
  + Loads and normalizes the input grayscale image.
  + Applies the selected morphological operation (mor\_op) in the chosen mode (manual or opencv).
  + If no specific operation is provided, applies and displays all operations in a grid layout.
  + Special handling **for Granulometry**: outputs numerical values to .txt and saves a corresponding line plot image.
  + Measures execution time and displays results.
* **main(argv)**
  + Parses command-line arguments using getopt.
  + Accepts the following arguments:
    - -i / --in\_file: Input image file path
    - -o / --out\_file: Output file path
    - -p / --mor\_operator: (Optional) Specific morphological operation to perform
    - -m / --mode: Processing mode (manual or opencv)
    - -t / --wait\_key\_time: Display wait time (in milliseconds)
  + Calls operator() with parsed parameters to execute the desired morphological processing workflow.

**How to run code:**

**1. Install Required Libraries**

First, make sure you have the necessary libraries installed, such as opencv, numpy, and morphological\_operator. You can install them using pip:

**pip install opencv-python numpy**

**Note**: The morphological\_operator library appears to be a custom-written library, so make sure it exists in the same directory or has been correctly installed.

**2. Command Line Structure**

Here’s the command to run the program from the command line:

**python main.py -i <input\_file> -o <output\_file> [-p <morph\_operator>] -m <mode> -t <wait\_key\_time>**

**3. Explanation of Parameters:**

* -i <input\_file>: Path to the input image file.
* -o <output\_file>: Path to save the result.
* -p <morph\_operator> (Optional): The specific morphological operation you want to apply (e.g., "Dilate", "Erode", etc.).
* -m <mode>: The execution mode ("manual" for custom-written algorithms or "opencv" for OpenCV).
* -t <wait\_key\_time> (Optional): Time to wait (in milliseconds) before closing the window displaying the image.

**4. Example:**

Suppose you have an image file input.jpg and you want to apply the "Dilate" operation using the custom algorithm (manual mode) and save the result to output.jpg. You would run:

**python main\_graysacle.py -i input.jpg -o output.jpg -p “Dilate” -m manual**

**5. Options:**

* If you don't specify a morphological operation, the program will apply all available operations.
* If you don't specify a wait time, the program will not wait and will automatically close the window displaying the image.

**6. Error Information:**

If there's an error, such as a missing image file or incorrect parameter, the program will show an appropriate error message.

**Image proof**:

**Grayscale.py**

**A computer screen with colorful text

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**A screen shot of a computer screen

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**main\_grayscale.py**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**III. Summarization of the usage**

**Detailed Usage and Algorithm Explanation:**

**GRAYSCALE IMAGE**

**1. Grayscale Dilation**

**A black text on a white background

Description automatically generated**

* **Usage**: Expands bright regions in a grayscale image by replacing each pixel with the maximum value in its neighborhood. Enhances bright structures.
* **Algorithm Explanation**:
  + Applies padding to the image to avoid boundary issues.
  + Iterates through the image, replacing each pixel with the maximum value within the structuring element.
  + Returns the dilated grayscale image.

**2. Grayscale Erosion**

**A math equations on a white background

Description automatically generated**

**Usage**: Shrinks bright regions by replacing each pixel with the minimum value in its neighborhood. Removes small bright details.

**Algorithm Explanation**:

* Applies padding to handle boundary effects.
* Iterates through the image, replacing each pixel with the minimum value within the structuring element.
* Returns the eroded grayscale image.

**3. Grayscale Opening**

**A black text with a white background

Description automatically generated**

* **Usage**: Removes small bright spots by first performing erosion (shrinking objects) followed by dilation (restoring shape). Helps with noise reduction.
* **Algorithm Explanation**:
  + Applies grayscale erosion to shrink bright areas.
  + Applies grayscale dilation to restore larger structures.
  + Returns the processed grayscale image.

**4. Grayscale Closing**

**A math symbols with a plus and a cross

Description automatically generated with medium confidence**

**Usage**: Fills small dark gaps by first performing dilation (expanding objects) followed by erosion (shrinking back). Helps close small holes in bright areas.

**Algorithm Explanation**:

* + Applies grayscale dilation to expand bright areas.
  + Applies grayscale erosion to restore original object shapes.
  + Returns the processed grayscale image.

**5. Grayscale Smoothing**

**A black math equation

Description automatically generated with medium confidence**

* **Usage**: Reduces noise and smooths object boundaries by applying both grayscale opening and closing sequentially.
* **Algorithm Explanation**:
  + First applies grayscale opening to remove small bright spots.
  + Then applies grayscale closing to fill small dark holes.
  + Returns the smoothed grayscale image.

**6. Morphology Gradient**

**A black text with a circle and a plus and a circle

Description automatically generated**

* **Usage**: Highlights object edges by computing the difference between dilated and eroded versions of the image. Enhances edge features.
* **Algorithm Explanation**:
  + Computes the difference: **(Dilation - Erosion)**.
  + Bright areas indicate regions with

**7. Top Hat**

**A black math equation

Description automatically generated with medium confidence**

* **Usage**: Extracts small bright details by subtracting the opened image from the original. Useful for enhancing fine structures.
* **Algorithm Explanation**:
  + Applies **grayscale opening** to remove small bright regions.
  + Subtracts the opened image from the original image, preserving only small bright details.
  + Returns the top-hat transformed image.

**8. Textual segmentation**

**A black and white text

Description automatically generated**

* **Usage**: Enhances texture features in an image by extracting fine structures using the top-hat transformation.
* **Algorithm Explanation**:
  + Uses **top-hat transformation** to highlight small bright details.
  + Enhances textural patterns by emphasizing variations in local brightness.
  + Returns the segmented image emphasizing textures.

**9. Granulometry**

* **Usage**: Analyzes the size distribution of bright structures by applying **grayscale opening** with different structuring element sizes.
* **Algorithm Explanation**:
  + Iterates through different kernel sizes.
  + Applies **grayscale opening** at each size to filter out smaller bright regions.
  + Computes the sum of pixels after each opening, indicating the presence of structures of different sizes.
  + Returns a list of summed pixel values for different kernel sizes, used to plot a granulometry curve.

**10. Reconstruction**

**A white text with black text

Description automatically generatedA math equations with black text

Description automatically generated with medium confidence**

* **Usage**: Restores missing or occluded image regions by iteratively applying **dilation** constrained by a mask.
* **Algorithm Explanation**:
  + Initializes a **marker** image, typically a subset of the mask.
  + Iteratively dilates the marker while ensuring it does not exceed the mask.
  + Stops when the marker no longer changes between iterations or reaches **max\_iter**.
  + Returns the reconstructed grayscale image.

**IV. EXPERIMENTS AND EVALUATION:**

**Grayscale image.**

**The results of granulometry function on landscape grayscale image**

A screenshot of a computer program

Description automatically generated

**A graph with a line

Description automatically generated**

**The Granulometry output file now contain different values**

granulometry function is now producing **three distinct values:**

* **46475241.000000**
* **46008343.000000**
* **45644549.000000**

**What do these values represent?**

Each value corresponds to the **sum of pixel intensities** after applying **Grayscale Opening** with different structuring element (SE) sizes.

* The first value (**46475241.000000**) corresponds to **SE size 3x3**.
* The second value (**46008343.000000**) corresponds to **SE size 5x5**.
* The third value (**45644549.000000**) corresponds to **SE size 7x7**.

**Why do the values decrease?**

As the structuring element **size increases**, **more small bright regions are removed**, leading to a **reduction in total pixel intensity**.

* **Smaller SEs (3x3)** only remove very small noise.
* **Larger SEs (7x7)** remove larger details and thin structures, reducing the total pixel sum.

**What does this tell us about the image?**

* The image **contains small bright structures** that are gradually removed as SE size increases.
* The decrease in values suggests **progressive removal of finer details**, which aligns with granulometry analysis.
* If values stayed **constant**, it would mean **no small structures were removed**, or SE sizes were **too small** to affect the image.

**The results of other custom grayscale morphological operations on landscape grayscale image**

**Opencv results**

**A collage of images of a river and trees

Description automatically generated**

**Manual results:**

**A collage of a river and a river

Description automatically generated**

**Comments:**

The custom grayscale morphological operations produce results that often differ from OpenCV's built-in functions, with some outputs appearing darker or less refined. OpenCV generally provides smoother and more accurate transformations due to optimized padding, normalization, and efficient kernel application. The main issues in the custom implementation likely stem from **inefficient pixel scaling, boundary handling, and loop-based computations**. Refining these aspects could improve accuracy and bring results closer to OpenCV’s optimized output.

**Comparison in general**

**Grayscale image operator custom vs opencv built in**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Custom Implementation** | **OpenCV Implementation** |
| **Performance (Speed)** | Slow (nested loops, sequential processing) | Fast (optimized C++ backend, SIMD, parallel processing) |
| **Accuracy** | Can be prone to errors (padding issues, structuring element misalignment) | Highly reliable and precise |
| **Dilation & Erosion** | Works if implemented correctly but slower | Optimized and fast |
| **Opening & Closing** | Affected by dilation/erosion accuracy | Standardized and efficient |
| **Hit-or-Miss** | May introduce errors if background processing is incorrect | Handles structuring elements correctly |
| **Boundary Extraction/ Gradient** | Accurate if erosion is properly done | Faster with correct results |
| **Region Filling** | Uses iterative dilation, slow for complex shapes | Uses flood fill, faster and more memory-efficient |
| **Thinning** | Works but may leave artifacts if conditions aren't checked properly | Optimized and standardizes Zhang-Suen or Guo-Hall algorithms |