Hw 2 Report - Image Processing

AU7009 Digital Image Processing, SJTU, 2022 Fall

By **Prof. L.S. Wang, T.Fang**

Table of Contents

Problem Specification

Implement and analyze the following image processing methods on a selected set of images (easy + complex + noisy):

- **[TASK 1] Gaussian Smoothing** (Gaussian Blur)
  - **implement**/test
  - demonstrate using kernels of size 5×5, 7×7, 11×11, 15×15, 19×19
  - observe & analyze:
    - effect of kernel sizes
- **[TASK 2-1] Sobel**: an operator for gradient-based edge-detection
  - **implement**/test

    - observe/analyze
        - thick boundaries
        - sensitivity to noise
        - boundary connectivity and region closure
- **[TASK 2-2] Laplacian**: an operator for 2nd-order gradient-based edge-detection
    - **implement**/test
    - observe/analyze
        - thick boundaries
        - sensitivity to noise
        - boundary connectivity and region closure
        - response to various gray scale gradients (especially w.r.t. details & non-boundary pixels)
- **[TASK 2-3] LoG** (Gaussian blur + Laplacian): an operator for gradient-based edge-detection
    - **implement**/test
    - observe/analyze
        - demonstrate using kernels (Gaussian blur kernels) of different sizes
        - suppression w.r.t. details and noise, using differnt kernel sizes
        - boundaries drifting, using differnt kernel sizes
        - response to small gray scale gradients, using differnt kernel sizes
        - double-boundary phenomenon of the zero-crossing
    - try
        - apply a high gradient threshold to remove fake boundaries
        - acquire single-pixel-wide boundaries
- **[TASK 2-4] Canny**: a pipeline for decent edge-detection
    - **implement**/test
    - observe/analyze
        - effect of lightening boundaries, by applying NMS (non-maximum suppression)
        - effect of highlighting boundaries, by applying double-thresholding
        - overall effect, by applying double-thresholding & single-low-thresholding & single-high-thresholding
        - overall effect, by applying Gaussian blur using kernels of different sizes
- **[TASK 3] Region Growing Algorithm**: an algorithm for edge-detection-related purposes
    - **implement**/test
    - observe/analyze
        - overall effect, by using different thresholds
        - overall effect of simple & complex images
- **[TASK 4] Split and Merge Algorithm**: an algorithm for edge-detection-related purposes
    - **implement**/test
    - observe/analyze
        - overall effect of simple & complex images

Environment

- OS: Windows 8.1 Pro (64-bit)
- Python 3.7.6

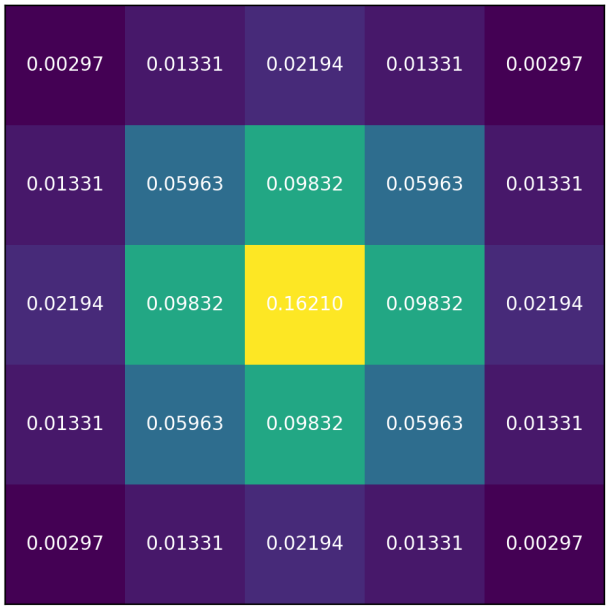
Implementation

The image chosen for implementation illustration is `1_gray-2.bmp`.

An implementation is given for each of the above methods, and will be compared with that of OpenCV.

TASK 1: Gaussian Smoothing - Implementation V.S. OpenCV

Original



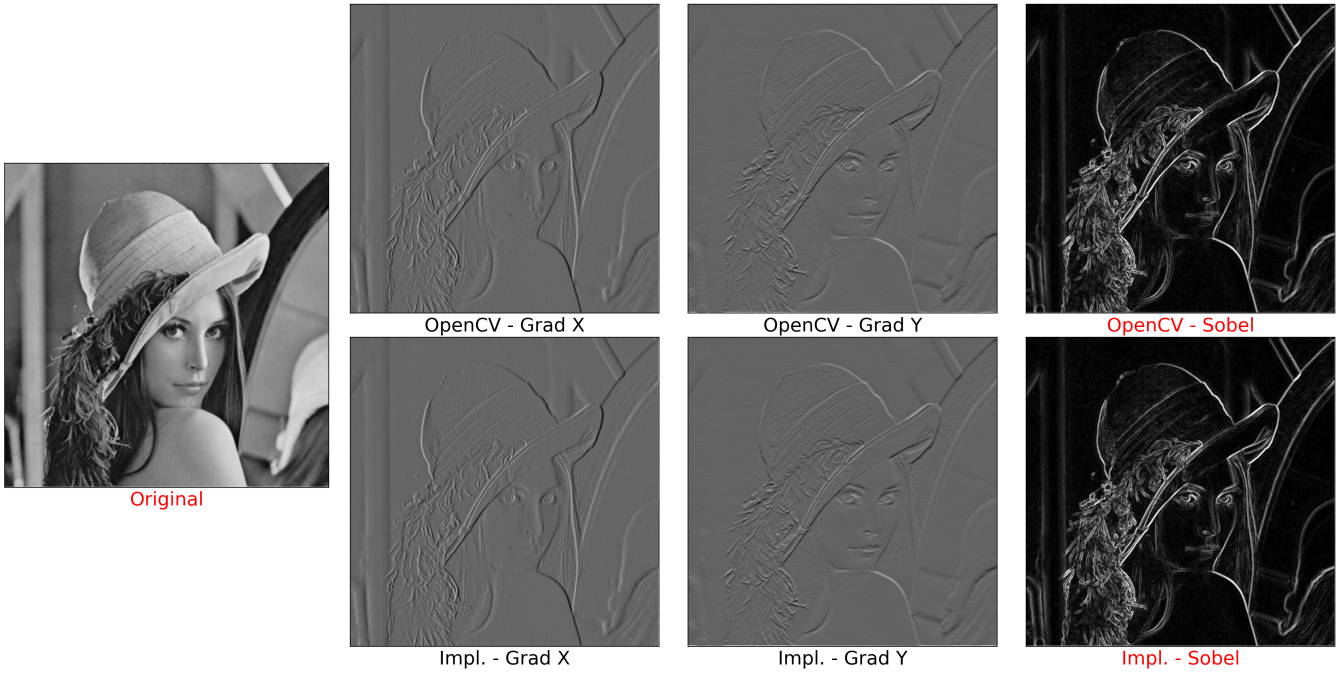
Kernel (Impl.) ($5 \times 5, \sigma = 1.0$)

Blurred (OpenCV) ($5 \times 5, \sigma = 1.0$)

Blurred (Impl.)

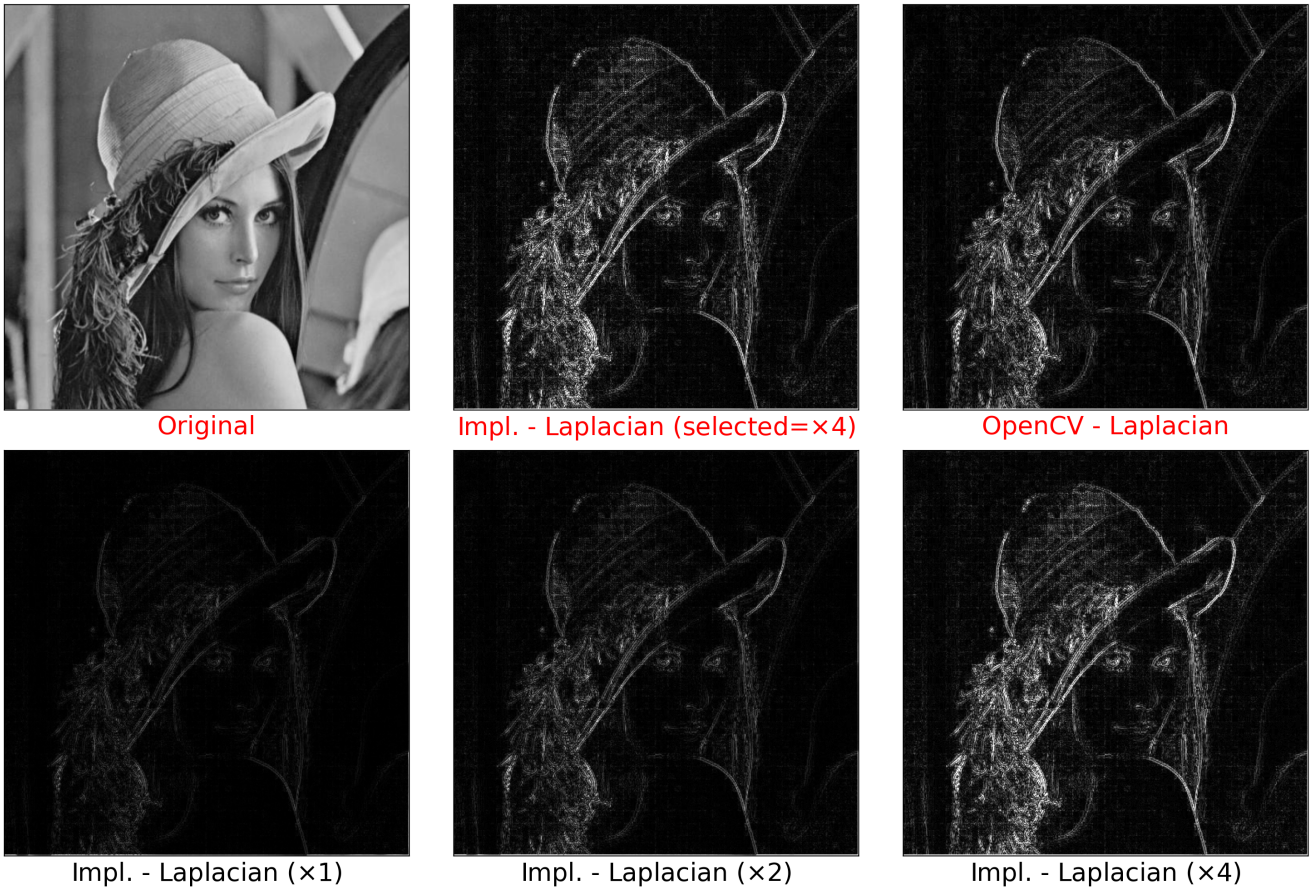
From the above demonstartion, we may observe that the implemented version behaves almost the same as that through direct OpenCV utilities call. Therefore, further expreiments will be comitted using OpenCV, to best utilize the features of this method.

TASK 2-1: Sobel - Implementation V.S. OpenCV



From the above demonstartion, we may observe that the implemented version behaves almost the same as that through direct OpenCV utilities call. Therefore, further expreiments will be comitted using OpenCV, to best utilize the features of this method.

TASK 2-2: Laplacian - Implementation V.S. OpenCV



From the above demonstartion, we may observe that the implemented version behaves almost the same as that through direct OpenCV utilities call. Therefore, further expreiments will be comitted using OpenCV, to best utilize the features of this method.

TASK 2-3: LoG - Implementation V.S. OpenCV


Original


OpenCV - Blurred ($3 \times 3, \sigma = 1.0$)


OpenCV - LoG


Impl. - Blurred ($3 \times 3, \sigma = 1.0$)

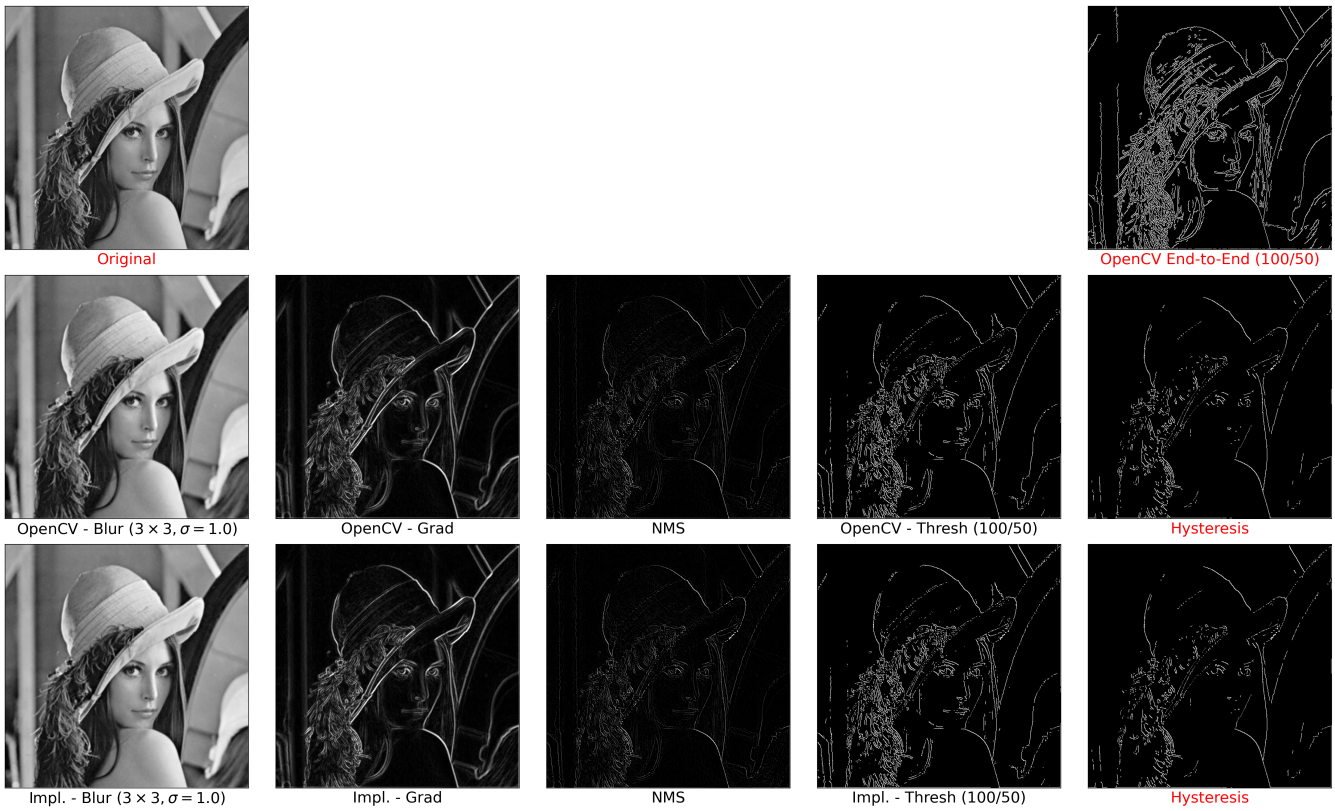

Impl. - LoG (Laplacian=×4)

From the above demonstartion, we may observe that the implemented version behaves almost the same as that through direct OpenCV utilities call. Therefore, further expreiments will be comitted using OpenCV, to best utilize the features of this method.

TASK 2-4: Canny - Implementation V.S. OpenCV

Note

- implementation of Canny consists of two groups:
  - labeled with prefix `OpenCV -` : if OpenCV provides utility calls, all are used (w.r.t., e.g., Gaussian blur, Sobel, thresholding, etc.)
  - labeled with prefix `Impl. -` : all operations are implemented
- double thresholding illustration is generated by:
  - setting strong boundaries as gray scale value `255`
  - setting weak boundaries as gray scale value `128`
  - setting all the other pixesl as gray scale value `0`



From the above demonstartion, we may observe that the implemented version behaves similar to that through direct OpenCV utilities call, but differently in the following aspects:

- result image of OpenCV end-to-end exhibits more details
  - `OpenCV End-to-End (100/50)` V.S. `Hysteresis`
  - even `OpenCV End-to-End (100/50)` V.S. `OpenCV - Thresh (100/50)` or `Impl. - Thresh (100/50)`

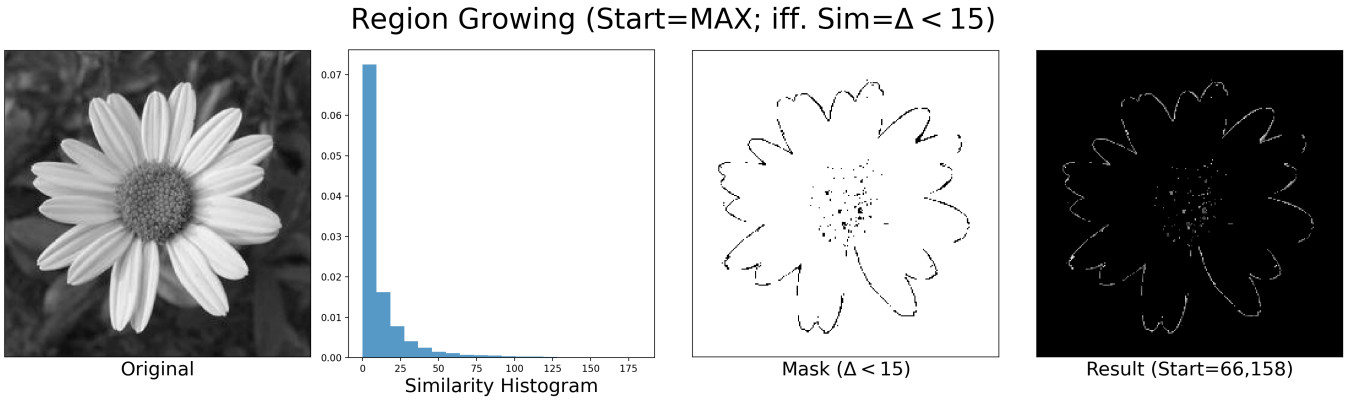
, which, might result from some unknown optimization adopted by the OpenCV's `Canny()`. However, considering the satisfactory enough results of the implemented version, we regard it a success.

Therefore, further expreiments will be comitted using OpenCV, to best utilize the features of this method.

TASK 3: Region Growing Algorithm - Implementation

Note

- `Start=MIN/MAX` means: the starting location is chosen automatically, as the first maximum/minimun point in gray scale values. The exact indexes are shown in figures.
- the similarity function is defined as: $|f(new) - f(old)|$, where,
  - $f(\cdot)$ extracts the gray scale value of the operand pixel
  - $|\cdot|$ is the absolute-value function
- region growing criteria is definded as:
  - for the current pixel p, reach out for each of its 8-adjacent neighbours p_i
  - calculate $\Delta = |f(p) - f(p_i)|$
  - add p_i to the grown region set iff. $\Delta < T_0$, where T_0 is the empirically set threshold

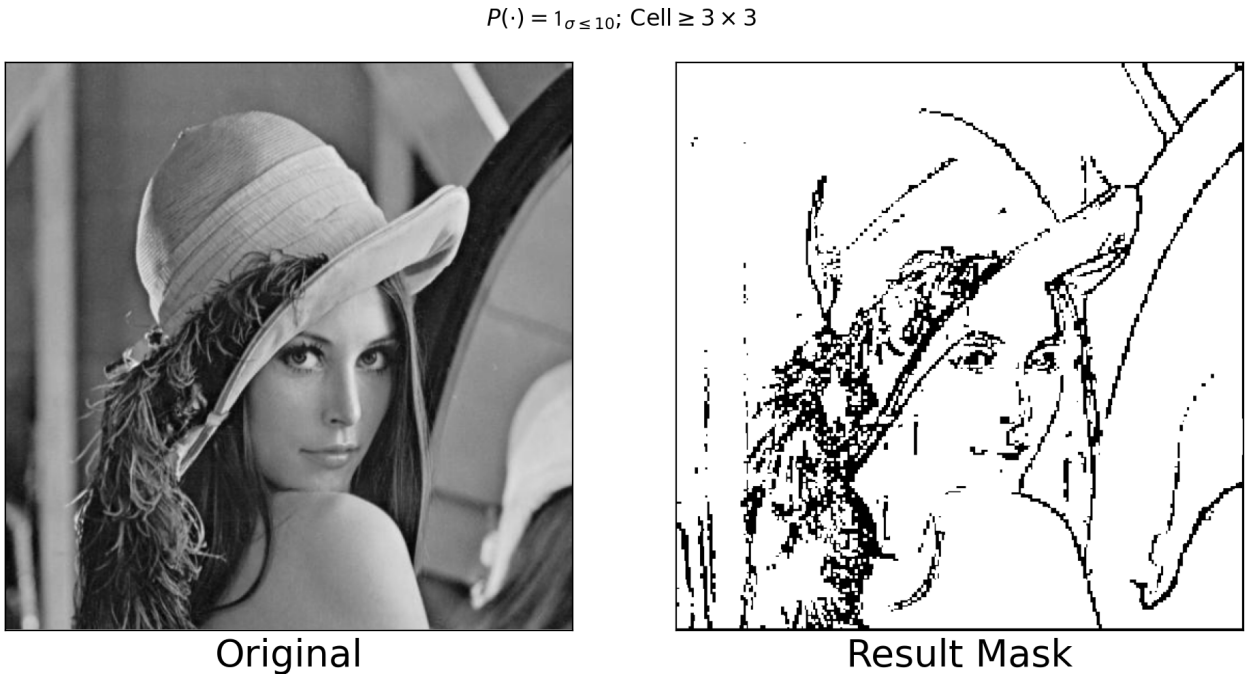



```
From the above demonstartion, we may observe that the implemented version behaves
satisfactory enough. Therefore, further expreiments will be comitted using such a
version.
```

TASK 4: Split and Merge Algorithm - Implementation

Note

- predicate P of a region R containing pixels $r_1, r_2, \cdots, r_{|R|}$ is defined as: $P(R) = 1_{std(\forall r_i) \leq T_0}$, where,
  - $std(\cdot)$ calculates the standard deviation
  - T_0 is an empirical threshold
- for merging (iff. $P(R) = True$), set the region as gray scale value 255



From the above demonstartion, we may observe that the implemented version behaves satisfactory enough. Therefore, further expreiments will be comitted using such a version.

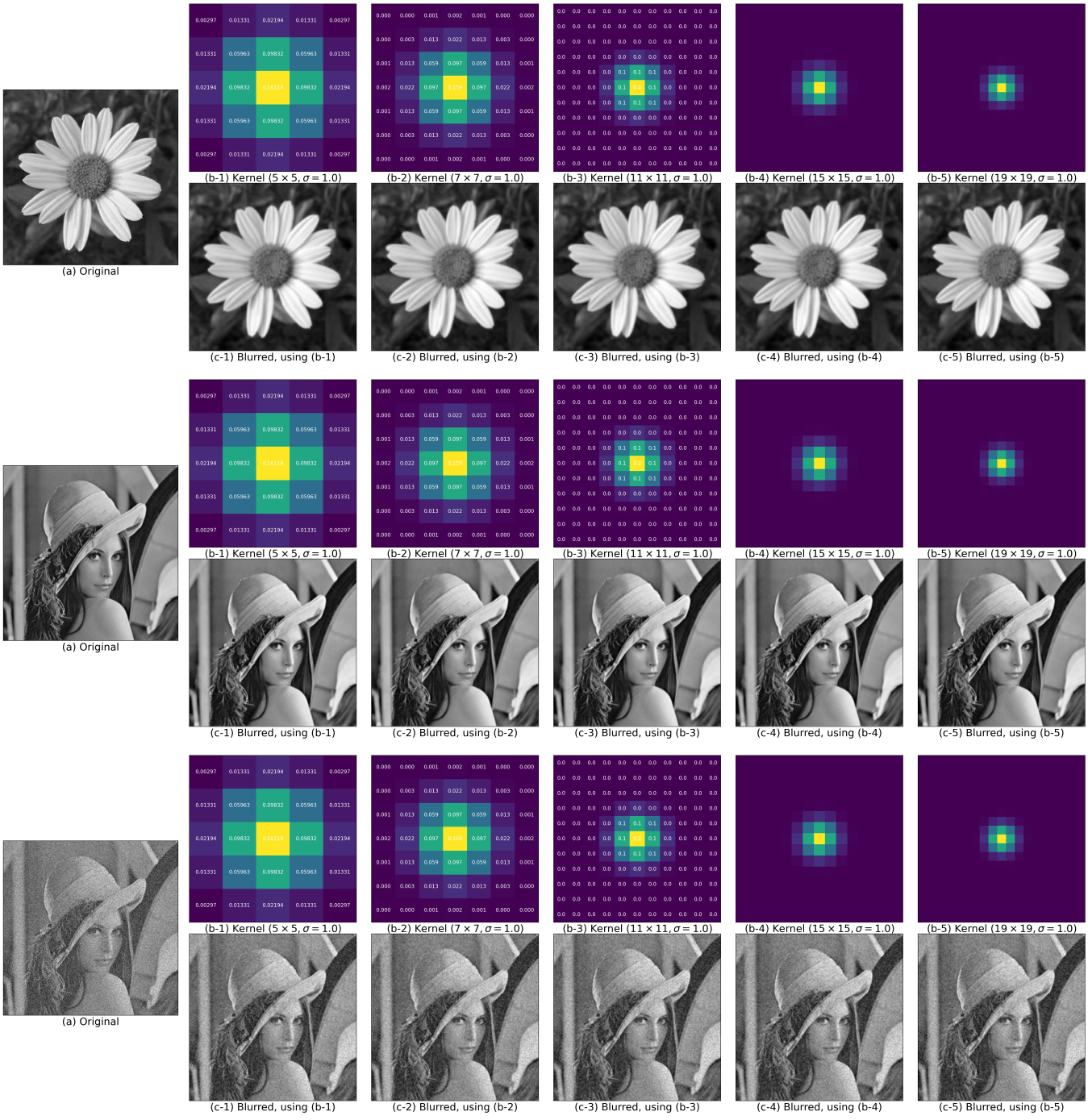
Experiment & Discussion

The images chosen for illustration are `简单图像3.jpg` (easy), `1_gray-2.bmp` (complex) and `gray高斯噪声3-1.jpg` (noisy).

TASK 1: Gaussian Smoothing

> --- Gaussian Blur

- demonstrate using kernels of size $5 \times 5, 7 \times 7, 11 \times 11, 15 \times 15, 19 \times 19$
- observe & analyze:
  - effect of kernel sizes



From the above figures, we may observe that

- generally speaking, the larger the kernel size, the fewer deatils are shown in the blurred result
- Guassian blur de-noise images with Gaussian noise well

TASK 2-1: Sobel

> --- an operator for gradient-based edge-detection

> - observe/analyze
> - thick boundaries
> - sensitivity to noise
> - boundary connectivity and region closure

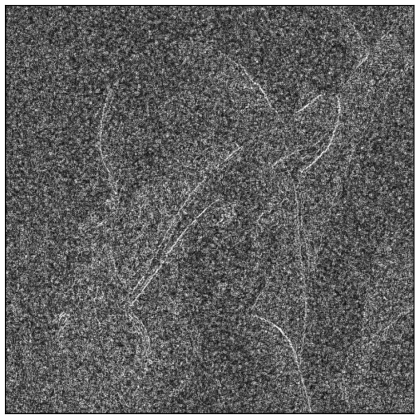

(a-1) Original


(a-2) Original


(a-3) Original


(b-1) Sobel of (a-1)


(b-2) Sobel of (a-2)


(b-3) Sobel of (a-3)

From the above figures, we may observe that

- explicit thick boundaries are shown
- Sobel is sensitive to noise (to an intermediate extent), as shown in the result of the image with Gaussian noise
- if without noise, boundaries are connected and are forming a good region closure

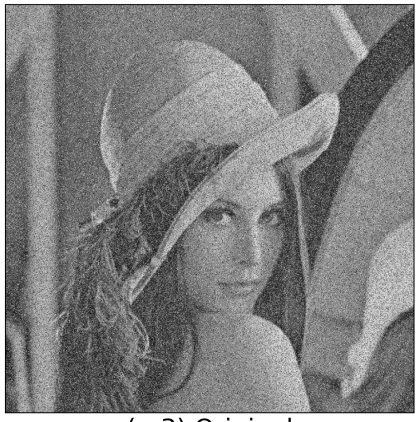
TASK 2-2: Laplacian

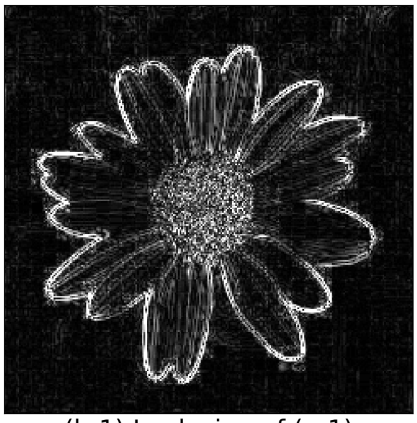
> --- an operator for 2nd-order gradient-based edge-detection

> - observe/analyze
> - thick boundaries
> - sensitivity to noise
> - boundary connectivity and region closure
> - response to various gray scale gradients (especially w.r.t. details & non-boundary pixels)


(a-1) Original


(a-2) Original


(a-3) Original


(b-1) Laplacian of (a-1)


(b-2) Laplacian of (a-2)


(b-3) Laplacian of (a-3)

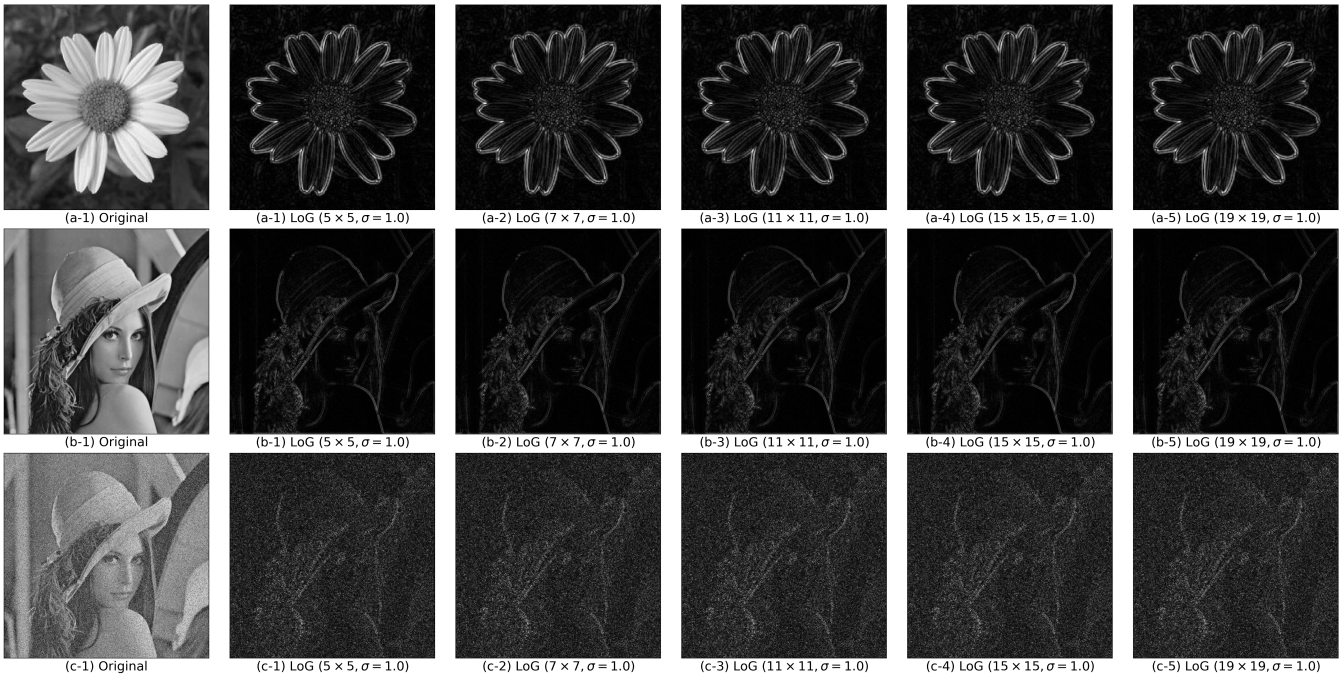
From the above figures, we may observe that

- explicit thick boundaries are shown
- Laplacian is extremely sensitive to noise (after all, it is 2nd-order-based), as shown in the result of the image with Gaussian noise
- if without noise, boundaries are connected and are forming a good region closure
- Laplacian suppresses weak gray scale gradients, based on observations: (1) boudaries (locations where gray scale gradients varies greatly) are less distinctive (lighter) than those of Sobel; (2) some "bakcground detailed boundaries" are gone.

TASK 2-3: LoG

> --- (Gaussian blur + Laplacian): an operator for gradient-based edge-detection

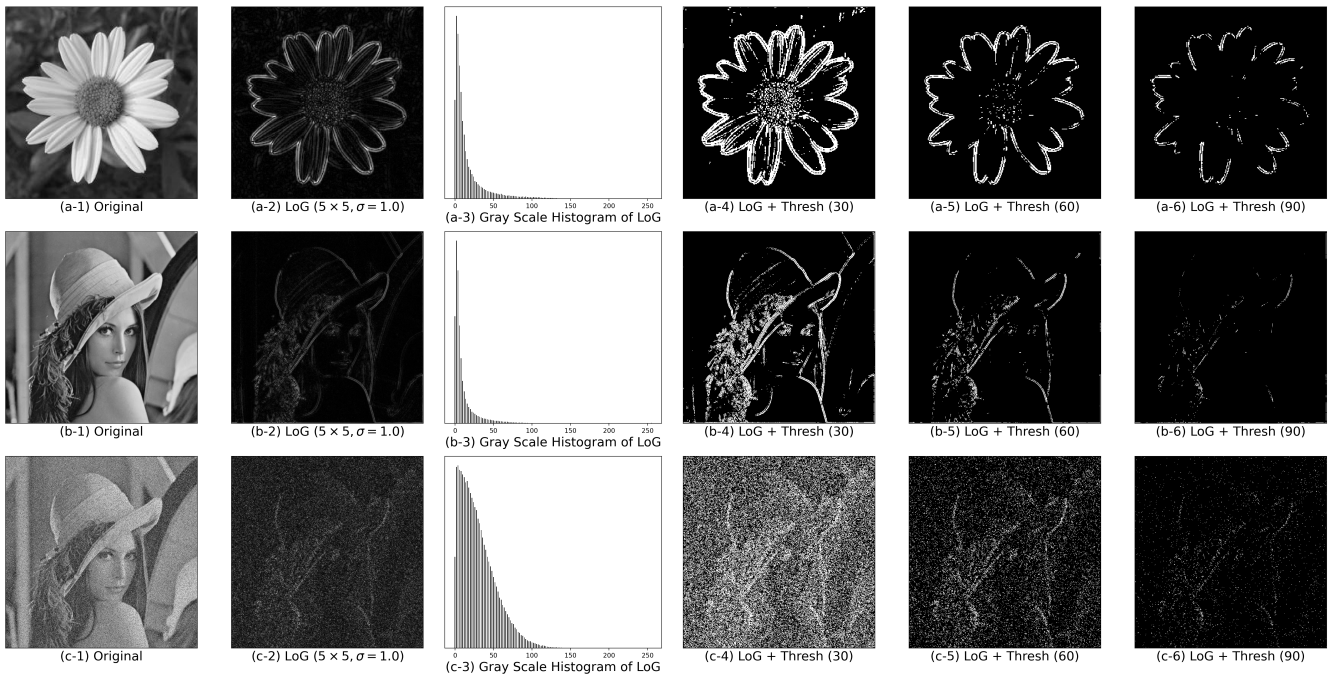
> - observe/analyze
> - demonstrate using kernels (Gaussian blur kernels) of different sizes
> - suppression w.r.t. details and noise, using differnt kernel sizes
> - boundaries drifting, using differnt kernel sizes
> - response to small gray scale gradients, using differnt kernel sizes
> - double-boundary phenomenon of the zero-crossing



From the above figures, we may observe that

- generally speaking, kernel sizes have little influence on the result, but slightly clearer details of main object's boundaries may be observed as the kernel sizes increase
- Gaussian blur suppresses (almost regardless of the kernel sizes)
    - details (small gray scale gradients)
    - the noise that is disastrous to Laplacian
- boundary drifting is unnoticeable
- double-boundary phenomenon is observed

> - try
>   - apply a high gradient threshold to remove fake boundaries

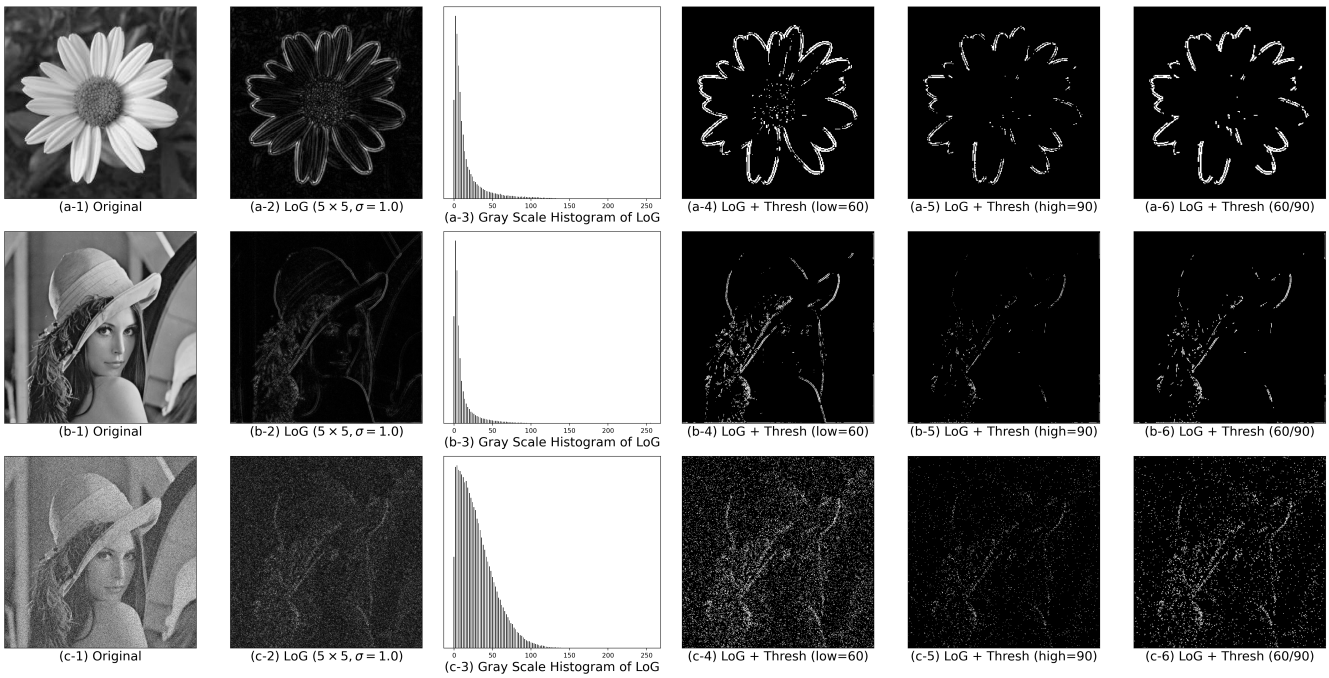


From the above figures, we may observe that

- by applying a higher gradient threshold, fake boundaries are suppressed at a greater extent than true ones, at the cost of diconnecting boundaries and losing details
- the higher the threshold is, the more distinct true boundaries are, and the fewer details are shown

> - try
>   - acquire single-pixel-wide boundaries

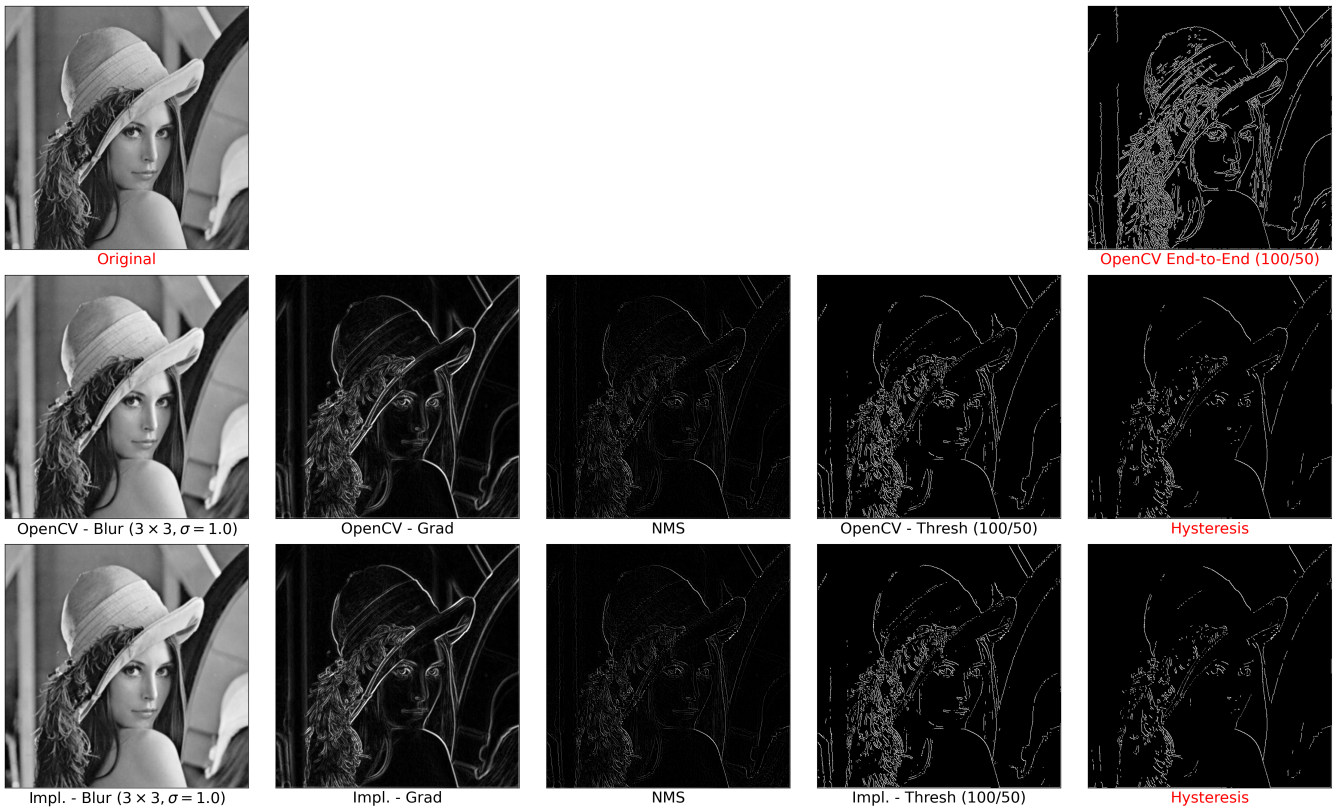
Therefore, motivated by Canny, we apply double-thresholding instead of single-thresholding, to try acquiring single-pixel-wide boundaries (& balance details, w.r.t. boundary- & non-boundart- pixels). The results (labeled as `(a/b/c-6) LoG + Thresh (%d/%d)`) are improved, compared with those of LoG only (labeled as `(a/b/c-2) LoG (...)`).



TASK 2-4: Canny

> --- a pipeline for decent edge-detection

> - observe/analyze
>   - effect of lightening boundaries, by applying NMS (non-maximum suppression)
>   - effect of highlighting boundaries, by applying double-thresholding



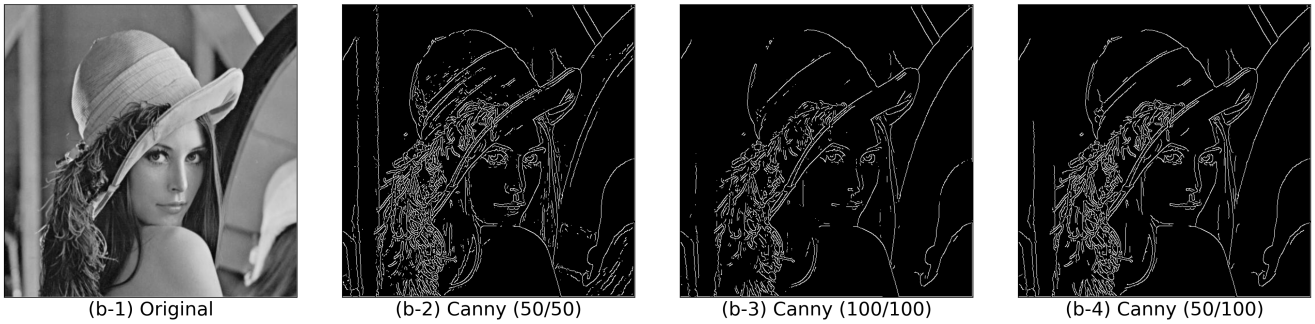
From the above figures (re-shown here, the same as those at the implementation demonstration), we may observe that

- NMS effectively suppresses insignificant boundaries
- double thresholding highlights the most significant boundaries (as strong ones, assigned gray scale value 255) and "suppresses" the significant but not the most significant ones (as weak ones, assigned gray scale value 128)

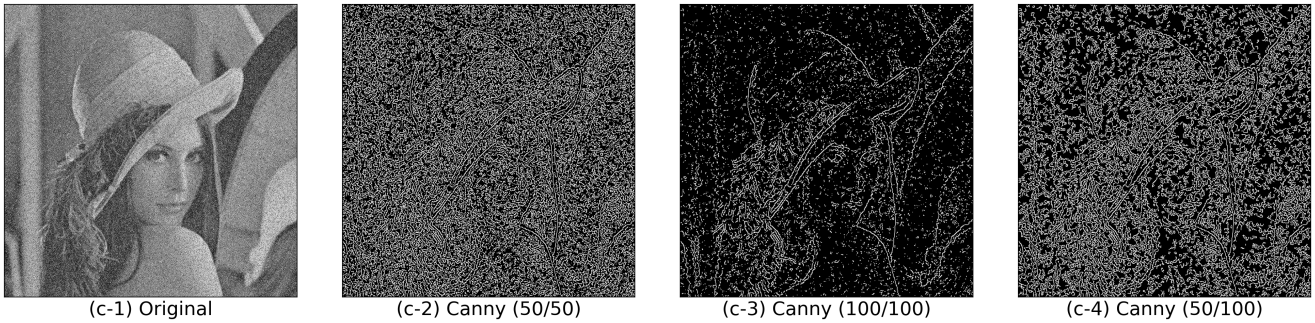
> - observe/analyze
>   - overall effect, by applying double-thresholding & single-low-thresholding & single-high-thresholding

Various Thresh Selection. Gaussian Blur $11 \times 11, \sigma = 1.0$

(a-1) Original (a-2) Canny (50/50) (a-3) Canny (100/100) (a-4) Canny (50/100)



(b-1) Original (b-2) Canny (50/50) (b-3) Canny (100/100) (b-4) Canny (50/100)

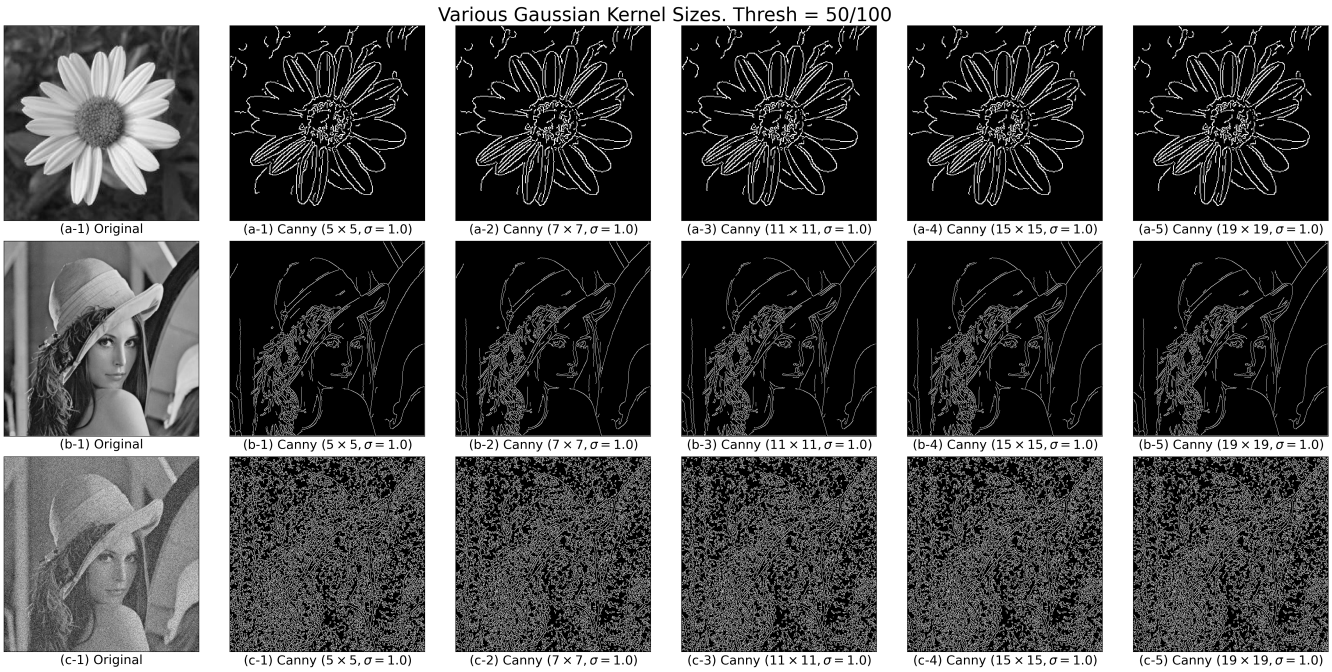


(c-1) Original (c-2) Canny (50/50) (c-3) Canny (100/100) (c-4) Canny (50/100)

From the above figures, we may observe that

- single-low-thresholding keeps the most details and suppresses the most the highlighting of main object's boundaries
- single-high-thresholding keeps the fewest details and suppresses the least the highlighting of main object's boundaries
- double-thresholding is a trade-off, which balancing details and the highlighting of main object's boundaries

> - observe/analyze
>   - overall effect, by applying Gaussian blur using kernels of different sizes



From the above figures, we may observe that

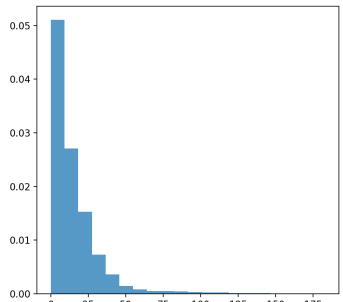
- kernel sizes have little influence on the overall effect
- slight details decrease may be observes with the increase of kernel sizes

TASK 3: Region Growing Algorithm

> --- an algorithm for edge-detection-related purposes

> - observe/analyze
>   - overall effect of simple & complex images


(a-1) Original, using Loc: MAX

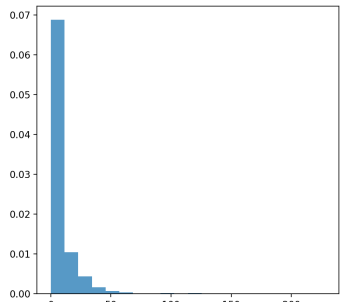


Similarity Histogram


Mask ($\Delta < 10$)


Result (Start=66,158)


(b-1) Original, using Loc: MIN

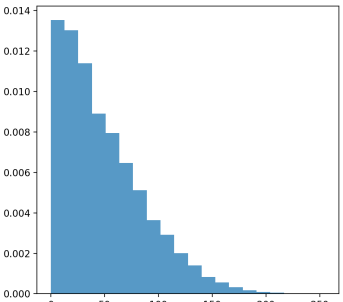


Similarity Histogram

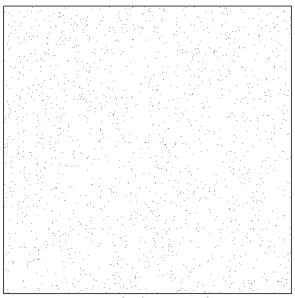

Mask ($\Delta < 10$)

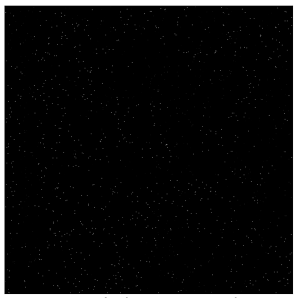

Result (Start=471,390)


(c-1) Original, using Loc: MIN



Similarity Histogram

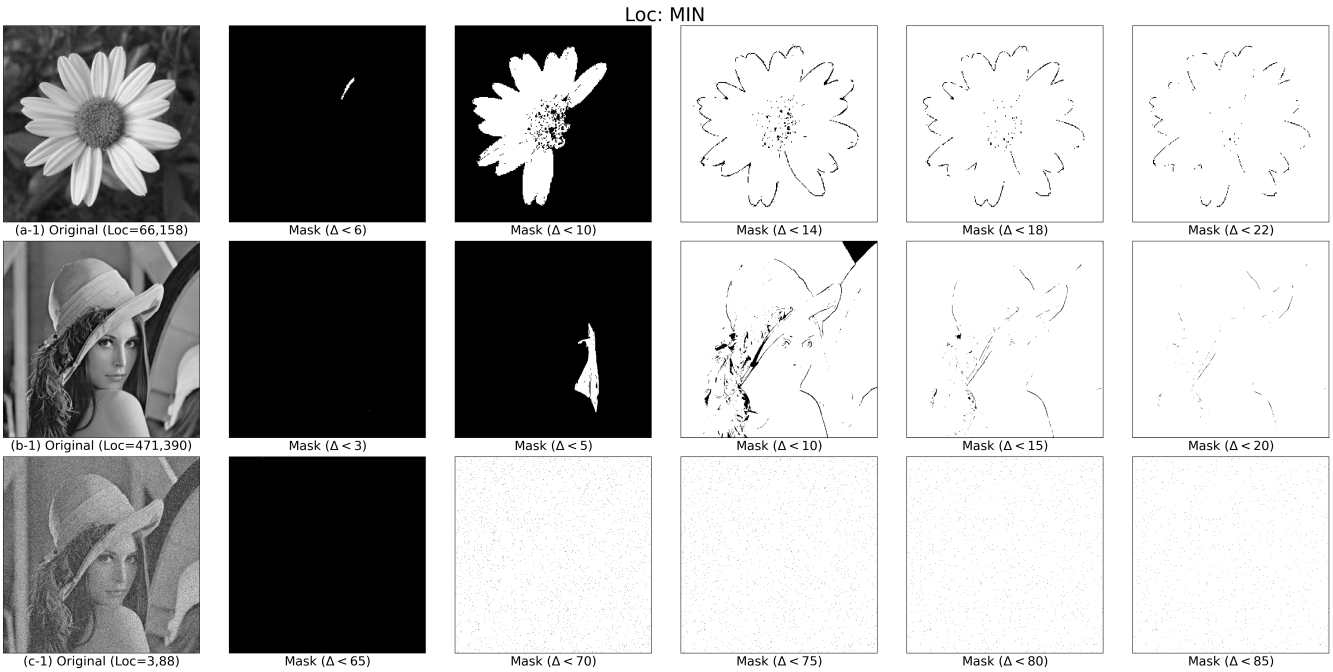

Mask ($\Delta < 75$)


Result (Start=3,88)

From the above figures, we may observe that

- the algorithm is sensitive to noise
- the algorithm is rather sensitive to the choice of similarity function and starting points, by observations
    - (simple image) only half of the flower is extracted
    - (complex image) significant boundaries are extracted, but not connected

> - observe/analyze
>   - overall effect, by using different thresholds



From the above figures, we may observe that

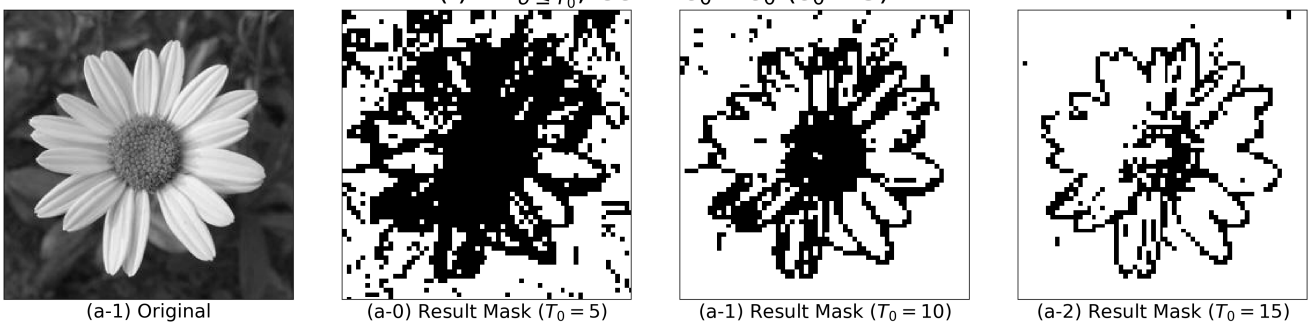
- choice of threshold, taking effect with the choice of similarity function, is dominant the overall performance
- with the current similarity function (abosulte difference of gray scale values), the larger the threshold, the less the boundaries are connected

TASK 4: Split and Merge Algorithm

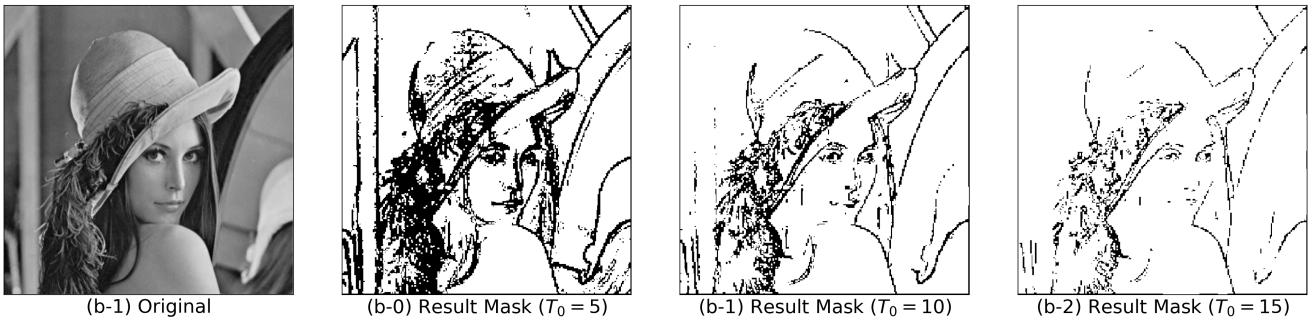
> --- an algorithm for edge-detection-related purposes

> - observe/analyze
>   - overall effect of simple & complex images

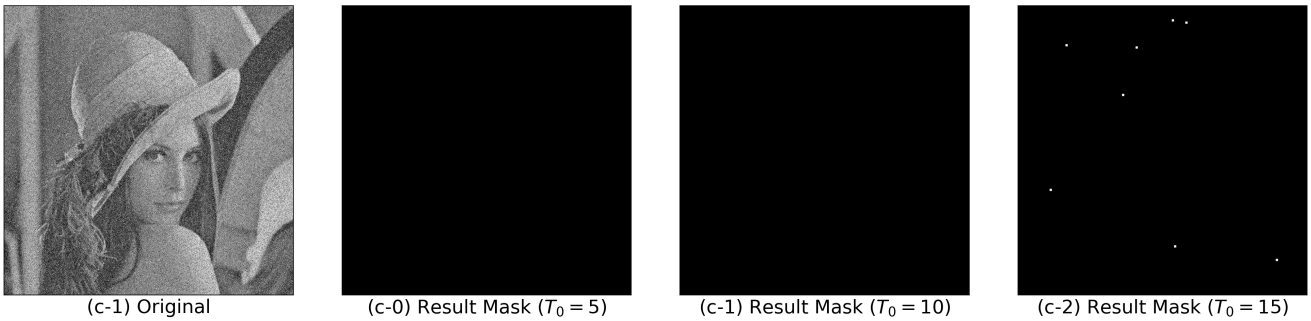
$P(\cdot) = 1_{\sigma \le T_0}$; Cell $\ge c_0 \times c_0$ ($c_0 = 3$)



(a-1) Original (a-0) Result Mask ($T_0 = 5$) (a-1) Result Mask ($T_0 = 10$) (a-2) Result Mask ($T_0 = 15$)



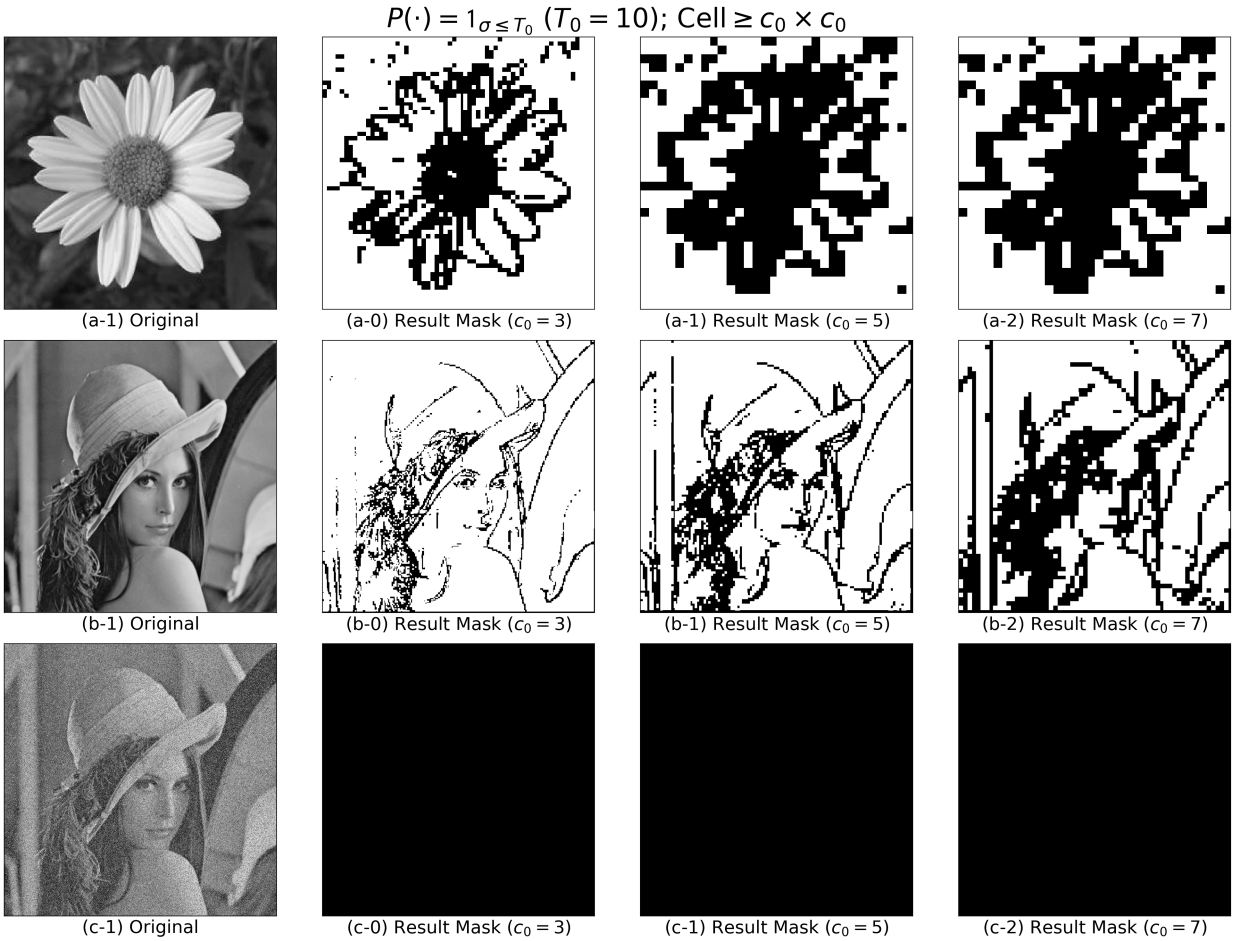
(b-1) Original (b-0) Result Mask ($T_0 = 5$) (b-1) Result Mask ($T_0 = 10$) (b-2) Result Mask ($T_0 = 15$)



(c-1) Original (c-0) Result Mask ($T_0 = 5$) (c-1) Result Mask ($T_0 = 10$) (c-2) Result Mask ($T_0 = 15$)

From the above figures, we may observe that

- the algorithm is sensitive to noise
- choice of threshold, taking effect with the choice of predicate, is dominant the overall performance
- with the current predicate (astandard deviation), the larger the threshold, the better the boundaries are extracted



From the above figures, we may observe that

- minimum sizes of regions have influence on the overall performance: the smaller the size, the more fine-grained the result

Appendix

Codes

Please check the codes shipped with the report.

Reference

- Gaussian filter - Wikipedia
- python - How to calculate a Gaussian kernel matrix efficiently in numpy? - Stack Overflow
- GaussianBlur() - OpenCV
- filter2D() - OpenCV
- Smoothing Images Examples - OpenCV
- Sobel operator - Wikipedia
- Sobel() - OpenCV
- Sobel Derivatives - OpenCV
- (Laplacian) Image Gradients - OpenCV
- Laplace Operator - OpenCV
- Laplacian() - OpenCV
- Canny Edge Detection Step by Step in Python - Towards Data Science

- OpenCV 区域生长算法（含代码）- CSDN
- c# - Image segmentation - Split and Merge (Quadtrees) - Stack Overflow
- 图像分割之区域分离 - CSDN