TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH 01
CÁC TOÁN TỬ HÌNH THÁI HỌC

Môn học: Ứng dụng xử lý ảnh số và video số

Họ tên: Huỳnh Lê Minh Nhật

MSSV: 1712632

I. GIỚI THIỆU CẤU TRÚC COMMAND:

Chương trình được thực thi bằng cmd với cấu trúc command có dạng:

python main.py -i <input_file> -o <output_file> -p <mor_operator> -t <wait_key_time> -f <input_text_file> -j <output_text_file>

Trong đó:

input_file: ảnh đầu vào

output_file: ảnh kết quả

mor_operator: toán tử cần thực hiện

wait_key_time: thời gian chờ

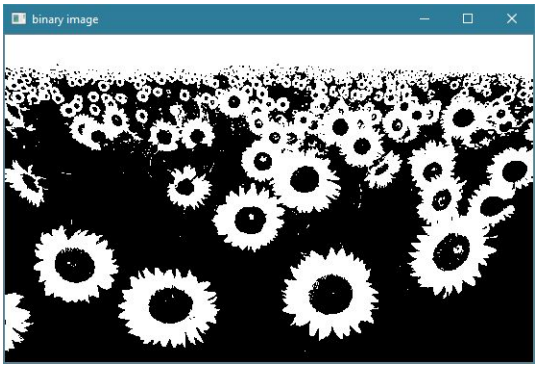
input_text_file: file txt đầu vào trong một số hàm

output_text_file: file txt kết quả trong một số hàm

II. TOÁN TỬ HÌNH THÁI HỌC NHỊ PHÂN:

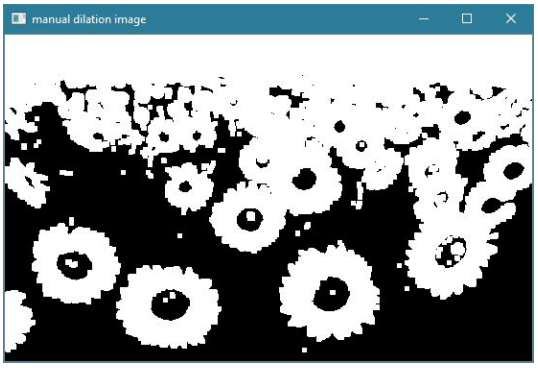
1. Binary Dilation:

- Command: python main.py -i test.jpg -o out.jpg -p dilate
- Kết quả:



Ảnh binary

OpenCV



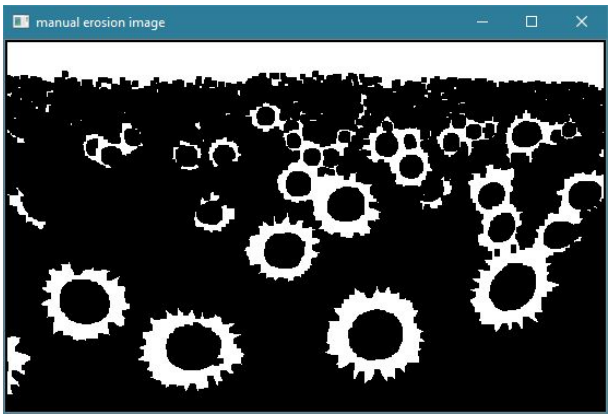
Manual

- Nhận xét: Kết quả của OpenCV và hàm tự cài đặt giống nhau

2. Binary Erosion:

- Command: python main.py -i test.jpg -o out.jpg -p erode
- Kết quả:

OpenCV

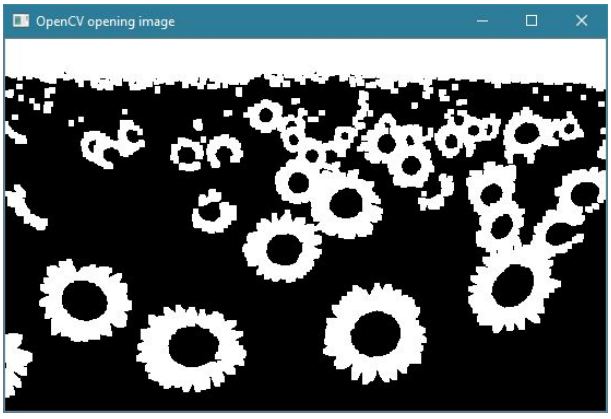


Manual

- Nhận xét: Kết quả của hàm tự cài đặt tương đồng với OpenCV, tuy nhiên có viền đen ở ngoài. Điều này là hợp lý vì theo lý thuyết thì những điểm ở biên sẽ không được bảo toàn

3. Binary Opening:

- Command: python main.py -i test.jpg -o out.jpg -p morph_open
- Kết quả:



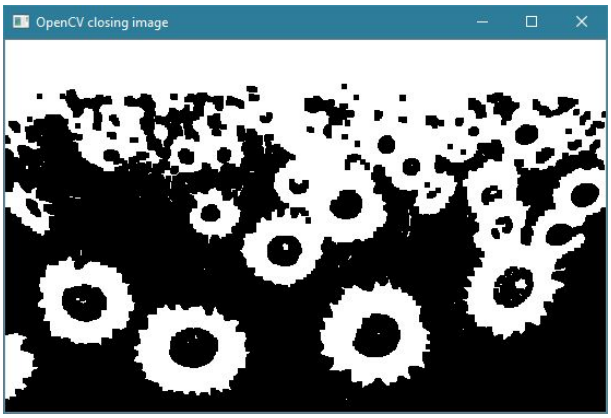
OpenCV

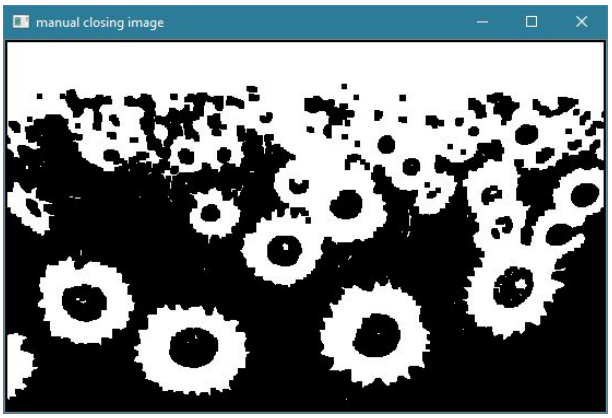
Manual

- Nhận xét: Mặc dù hàm erode manual khác với OpenCV nhưng do có toán tử dilate phía sau nên kết quả cuối cùng tương đồng nhau

4. Binary Closing:

- Command: python main.py -i test.jpg -o out.jpg -p morph_close
- Kết quả:



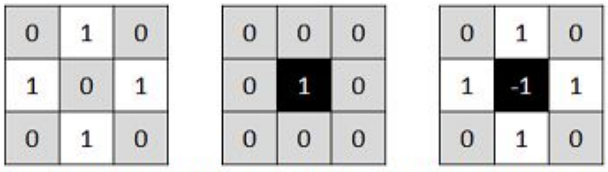


OpenCV Manual

- Nhận xét: Kết quả manual xuất hiện viền đen do phép erode, các phần còn lại đều giống với OpenCV

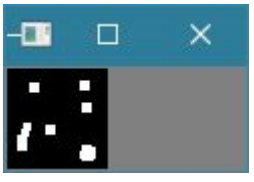
5. Hit-or-Miss:

- Hàm hit-or-miss của OpenCV không dùng 2 kernel như lý thuyết mà chỉ dùng 1 kernel gộp chung với kí hiệu 1: foreground, -1: background, 0: không quan tâm

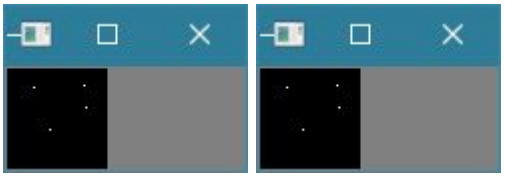


Structuring elements (kernels). Left: kernel to 'hit'. Middle: kernel to 'miss'. Right: final combined kernel

- Do đó ta phải biến đổi kernel 5x5 ban đầu thành 7x7 với các phần tử viền có giá trị -1
- Để thuận tiện cho việc tái sử dụng thì hàm hit-or-miss manual cũng được thay đổi ở parameter: chỉ nhận vào 1 kernel gộp. Trong hàm sẽ có các câu lệnh tách kernel gộp thành 2 kernel ứng với công thức lý thuyết
- Command: python main.py -i hitmiss.png -o out.jpg -p hitmiss
- Kết quả:



Ảnh gốc



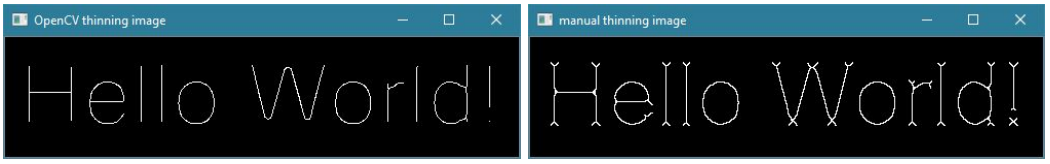
OpenCV Manual

- Nhận xét: Kết quả 2 hàm giống nhau, với chấm trắng là vị trí hình vuông 5x5 (không có phần thừa)

6. Thinning:

- Command: python main.py -i text.png -o out.jpg -p thinning
- Kết quả:

Ảnh gốc

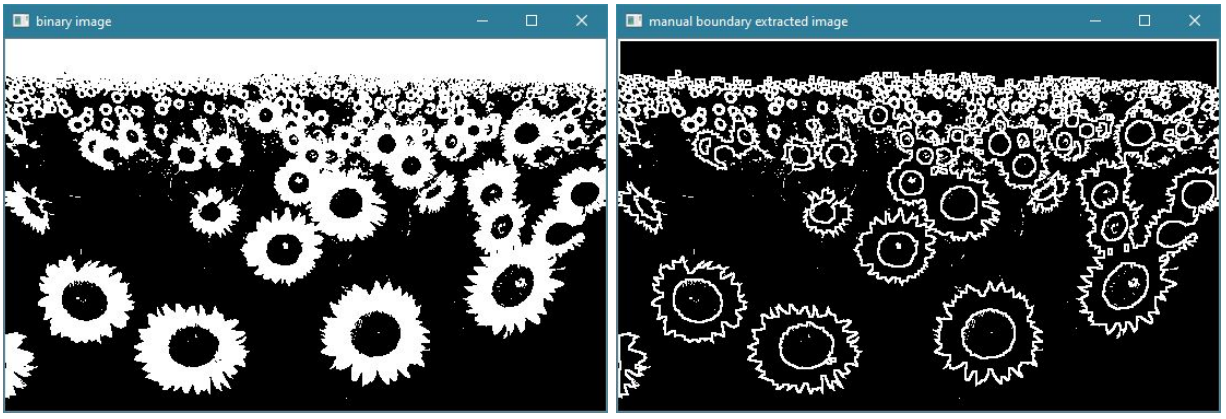


OpenCV Manual

- Nhận xét: Các nét chính ở 2 ảnh kết quả tương đồng nhau, tuy nhiên ở ảnh tự cài đặt thì nét vẫn dày hơn và ở đầu các nét bị loe ra

7. Boundary Extraction:

- Command: python main.py -i test.jpg -o out.jpg -p boundary
- Kết quả:



Ảnh gốc Boundary extraction

- Nhận xét: Ảnh kết quả tách biên khá tốt

8. Hole Filling:

- Command: python main.py -i holes.png -o out.jpg -p holefilling -f hole.txt
- Hole.txt là file text chứa toạ độ các điểm khởi tạo, mỗi điểm nằm trên 1 dòng

- Ví dụ:

  Hole.txt

  9 5

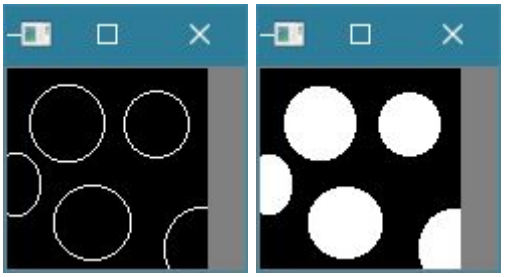
  9 28

  23 5

  25 19

  20 21

- Kết quả:



Ảnh gốc Ảnh filling

- Nhận xét: Các chỗ trống được lấp đầy, tuy nhiên thời gian thực thi rất chậm và có hạn chế là người dùng phải xác định điểm khởi tạo để lấp vùng

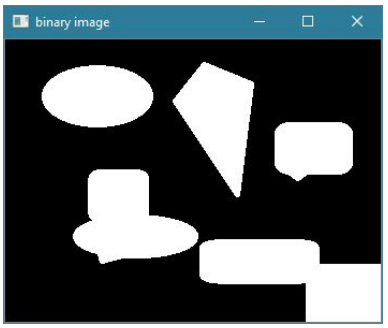
9. Extraction of Connected Components:

- Command: python main.py -i components.png -o out.jpg -p extcomponent -j out.txt
- Trong đó:

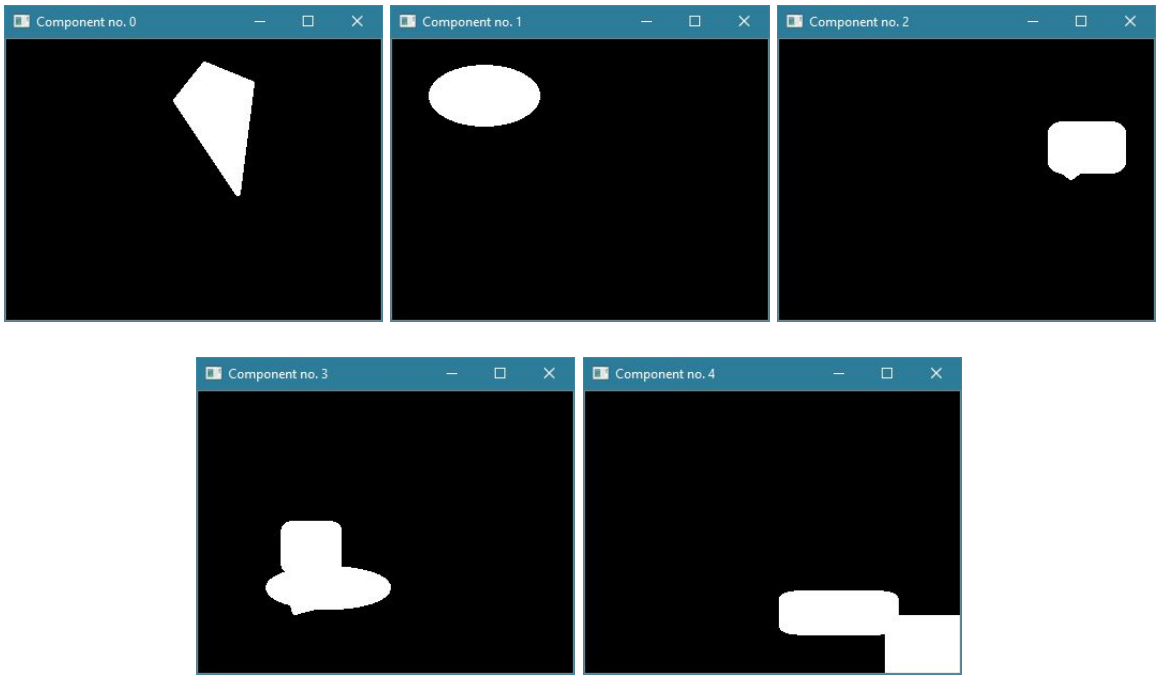
out.jpg là tên chung cho các thành phần liên thông, khi xuất kết quả sẽ tự động thêm hậu tố là số thứ tự. Ví dụ: out_0.jpg, out_1.jpg

out.txt là file text chứa thông tin các thành phần liên thông gồm số thứ tự, tổng pixel của mỗi thành phần

- Kết quả:



Ảnh gốc



Các thành phần liên thông

Out.txt

Component no. 0 contains: 5327 pixels

Component no. 1 contains: 4739 pixels

Component no. 2 contains: 3505 pixels

Component no. 3 contains: 6345 pixels

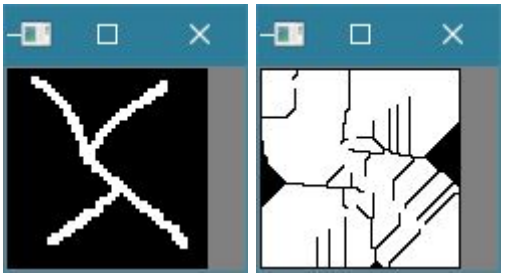
Component no. 4 contains: 8126 pixels

- Nhận xét: Ảnh được tách thành các thành phần liên thông khá tốt nhưng thời gian thực thi còn chậm

10. Thickening:

- Command: python main.py -i thick.png -o out.jpg -p thickening

- Kết quả:

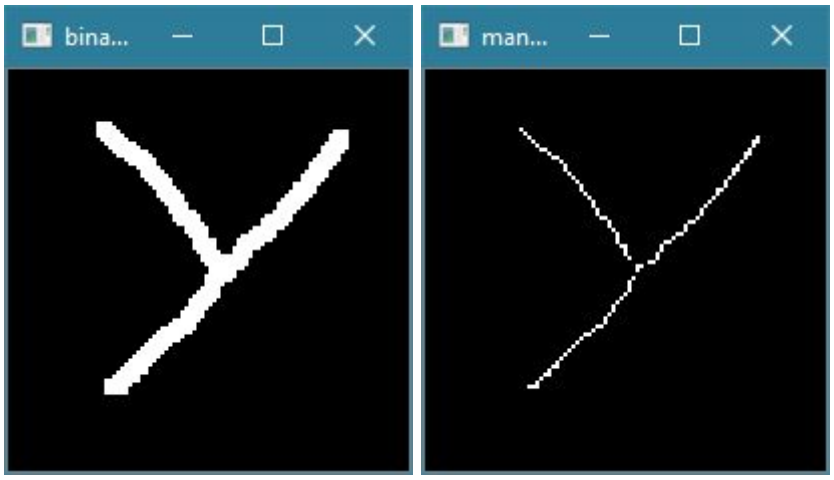


Ảnh gốc Ảnh thickening

- Nhận xét: Các nét chính đã được dày lên và vẫn giữ được cấu trúc của đối tượng, tuy nhiên vẫn còn những đường vân không mong muốn

11. Skeletons:

- Command: python main.py -i skeleton.png -o out.jpg -p skeleton
- Kết quả:



- Nhận xét: Ảnh thể hiện được khung xương của vật thể, tuy nhiên vẫn còn những điểm chưa liền mạch

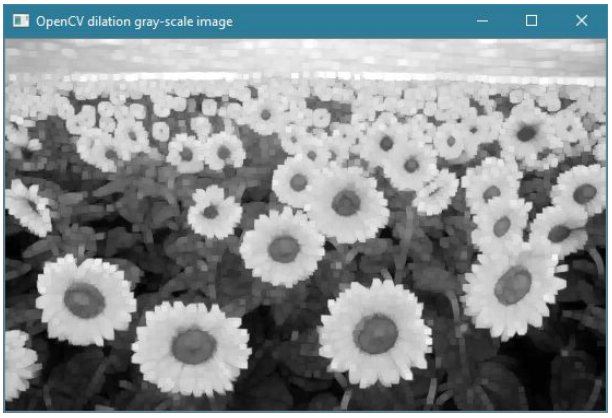
III. TOÁN TỬ HÌNH THÁI HỌC VỚI ẢNH ĐỘ XÁM:

1. Dilation:

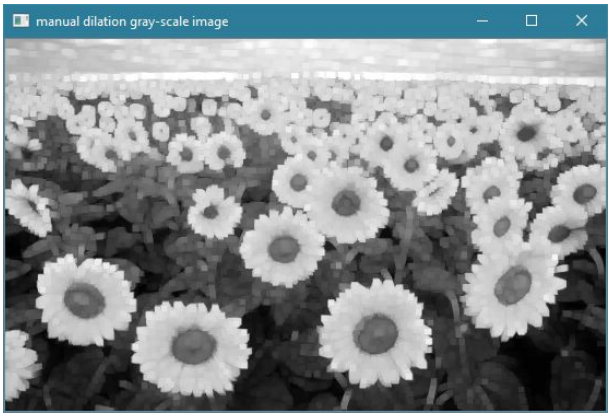
- Các hàm manual được cài đặt theo lý thuyết nên có thể dùng với kernel bất kỳ, trong khi các toán tử trong OpenCV chỉ dùng kernel phẳng.
- Kernel phẳng: $$[f \oplus b](x, y) = \max_{(s,t) \in b} \{f(x - s, y - t)\}$$
- Kernel không phẳng: $$[f \oplus b_N](x, y) = \max_{(s,t) \in b_N} \{f(x - s, y - t) + b_N(s, t)\}$$
- Do đó để đảm bảo kết quả, ta sẽ dùng kernel toàn 0 cho hàm manual
- Command: python main.py -i test.jpg -o out.jpg -p dilate_gray

- Kết quả:

Ảnh gốc



OpenCV

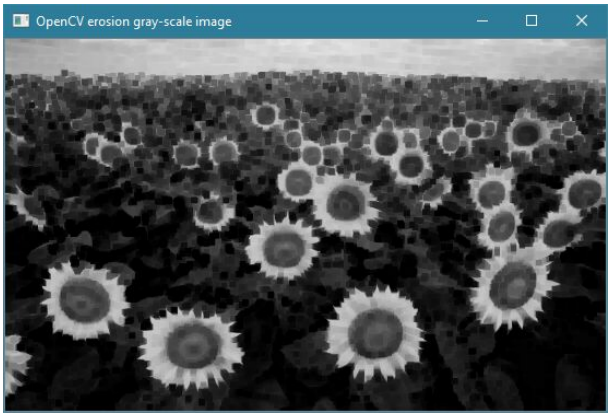


Manual

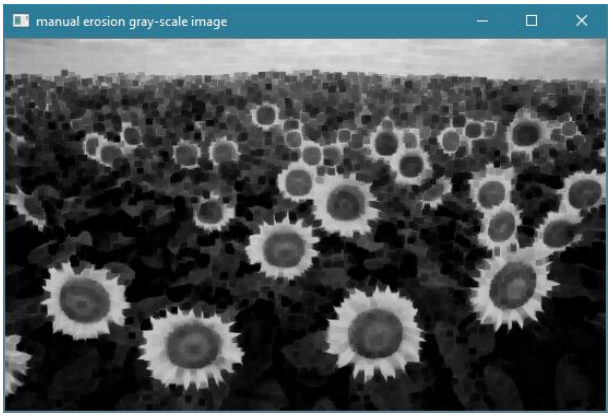
- Nhận xét: Hai hàm trả về kết quả giống nhau

2. Erosion:

- Command: python main.py -i test.jpg -o out.jpg -p erode_gray
- Kết quả:



OpenCV

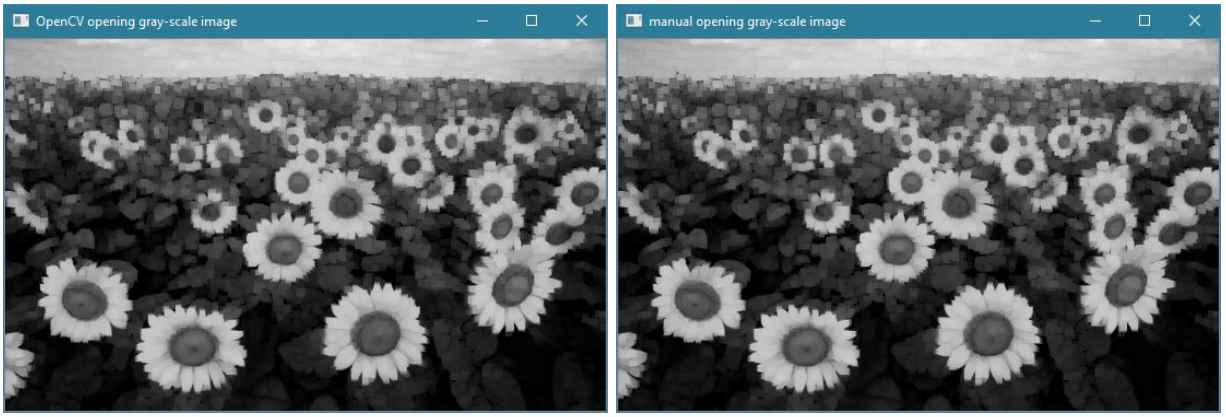


Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

3. Opening:

- Command: python main.py -i test.jpg -o out.jpg -p morph_open_gray
- Kết quả:

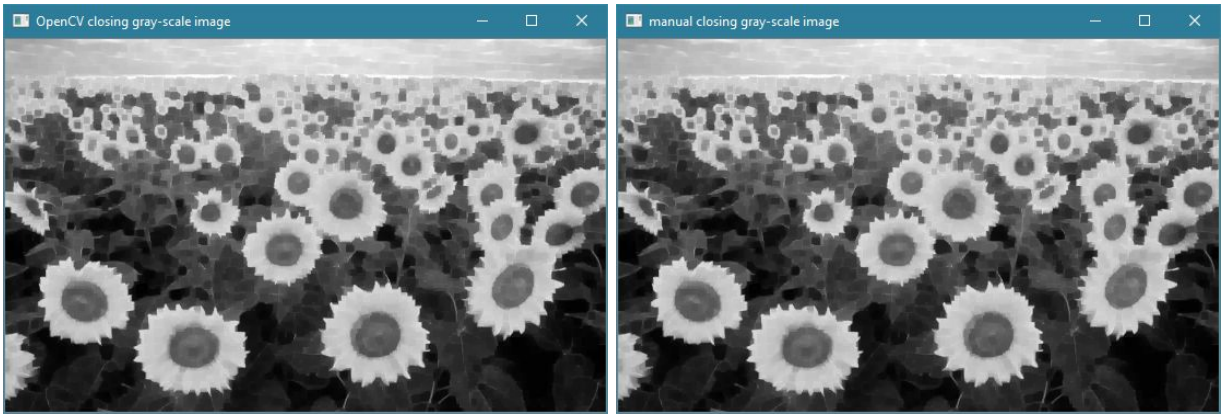


OpenCV Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

4. Closing:

- Command: python main.py -i test.jpg -o out.jpg -p morph_close_gray
- Kết quả:

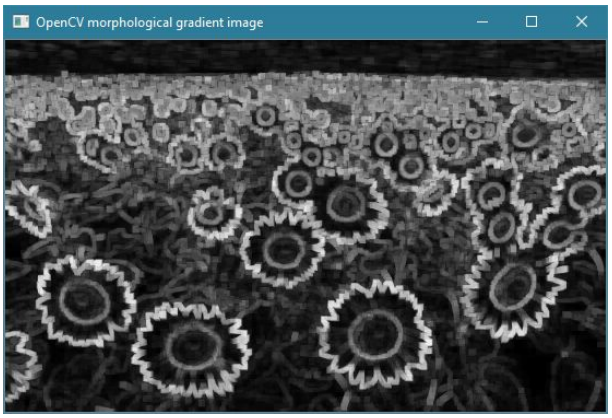


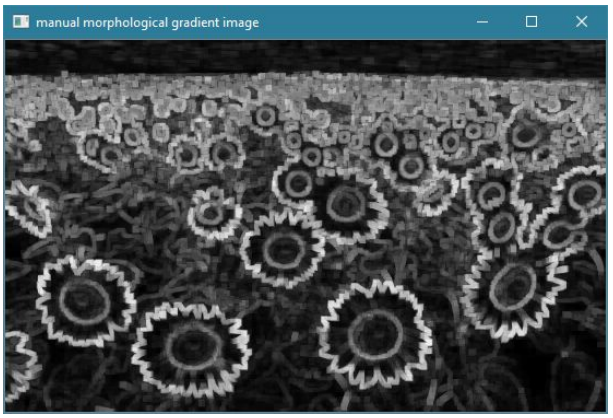
OpenCV Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

5. Morphological Gradient:

- Command: python main.py -i test.jpg -o out.jpg -p gradient
- Kết quả:





OpenCV Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

6. Top-Hat:

- - Command: python main.py -i test.jpg -o out.jpg -p tophat
- Kết quả:

OpenCV Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

7. Black-Hat:

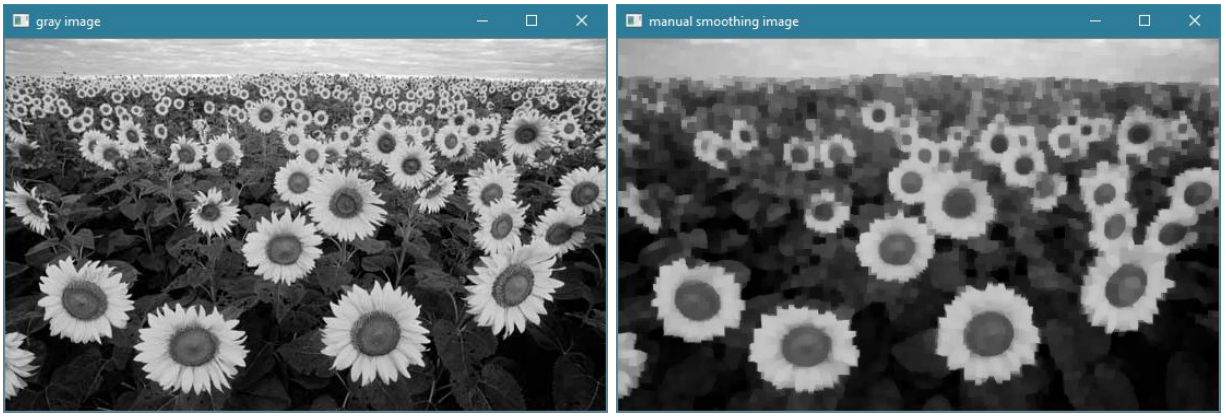
- Command: python main.py -i test.jpg -o out.jpg -p blackhat
- Kết quả:

OpenCV Manual

- Nhận xét: Hai hàm trả về kết quả giống nhau

8. Smoothing:

- Command: python main.py -i test.jpg -o out.jpg -p smoothing
- Kết quả:

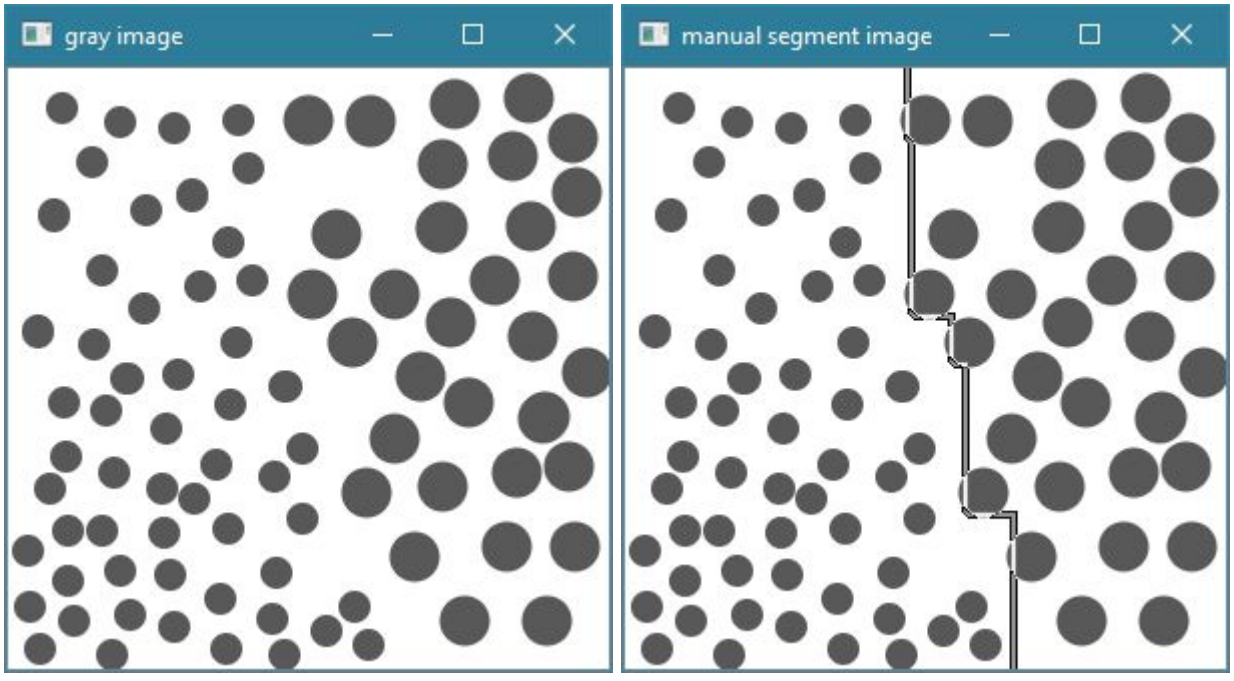


OpenCV Manual

- Nhận xét: Ảnh được làm trơn khá tốt

9. Textural Segmentation:

- Command: python main.py -i texture.png -o out.jpg -p segment
- Kết quả:



Ảnh gốc Ảnh segment

- Nhận xét: Ảnh đã được segment, tuy nhiên đường phân chia vẫn còn đè lên một số phần tử. Mặc khác, việc lựa chọn kernel ảnh hưởng rất lớn đến kết quả. Ở đây chọn kernel 1 là 25x25, kernel 2 là 100x100. Ta có thể chọn các kernel khác nhau để có kết quả tối ưu hơn

IV. Đánh giá mức độ hoàn thành:

- Cài đặt được các toán tử cơ bản và một số toán tử mở rộng
- Các toán tử chưa hoàn thành và chưa cài đặt: Convex Hull, Pruning, Morphological Reconstruction (ảnh xám và nhị phân), Granulometry

HẾT