ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH

Môn: Ứng dụng Xử lý ảnh số & video số

Lab 01: CÁC TOÁN TỬ HÌNH THÁI HỌC

Giảng viên hướng dẫn:
Lý Quốc Ngọc
Nguyễn Mạnh Hùng

Sinh viên thực hiện:

Nguyễn Quí Em
MSSV: 1712399

Ngày 31 tháng 05 năm 2020

MỤC LỤC

NỘI DUNG

Video demo: https://www.youtube.com/watch?v=rlRGTEcXclk

1. Các toán tử hình thái học trên ảnh nhị phân.

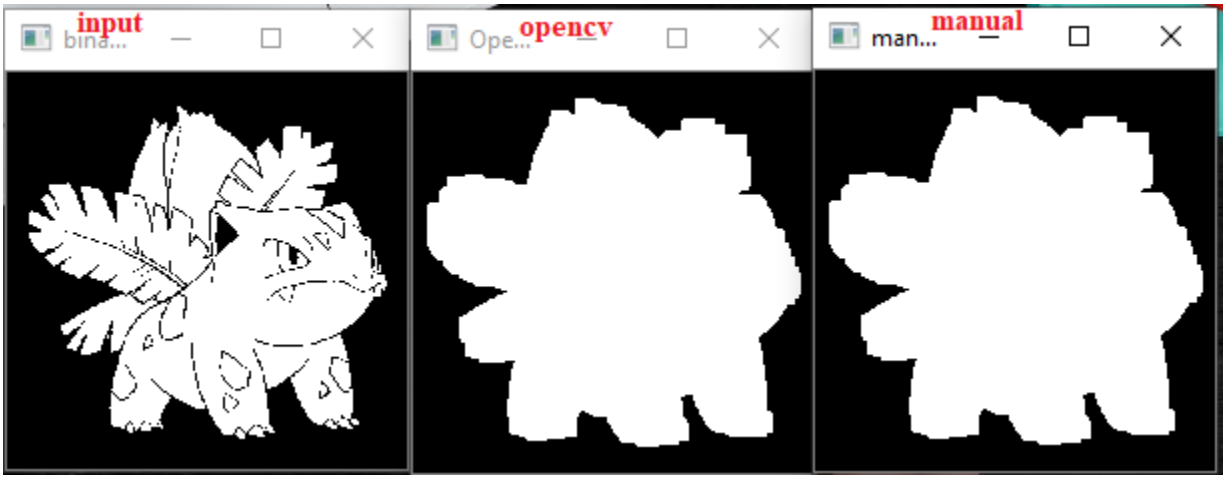
Erosion

Câu lệnh: main.py -i <input file> -o <output file> -p **erode** -t <wait key time>

Nhận xét: Kết quả chính xác tương tự OpenCV.

Dilation

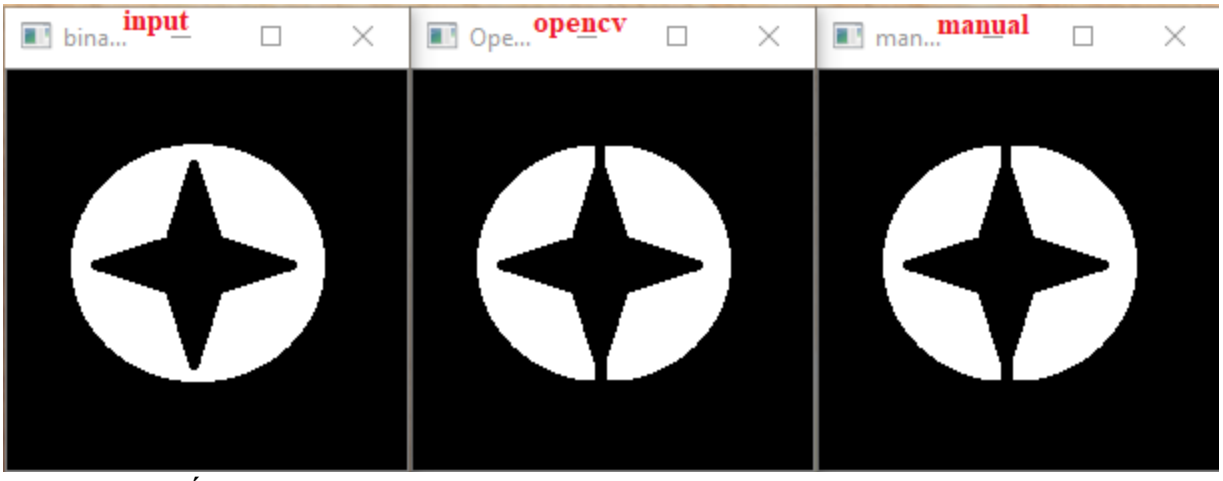
Câu lệnh: main.py -i <input file> -o <output file> -p **dilate** -t <wait key time>



Nhận xét: Kết quả chính xác tương tự OpenCV.

Opening

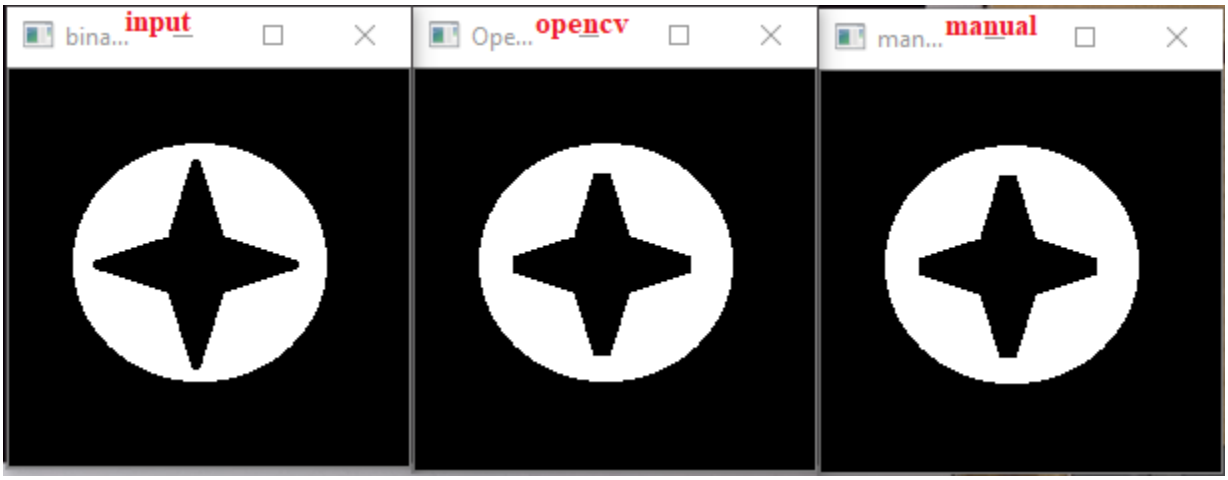
Câu lệnh: main.py -i <input file> -o <output file> -p **open** -t <wait key time>



Nhận xét: Kết quả chính xác tương tự OpenCV.

Closing

Câu lệnh: main.py -i <input file> -o <output file> -p **close** -t <wait key time>



Nhận xét: Kết quả chính xác tương tự OpenCV.

Hit-or-miss

Quy tắc: Kernel truyền vào được tạo sinh theo quy tắc như hình vẽ (final combined kernel). Sau đó hàm hitmiss sẽ phân tách kernel này thành kernel hit và kernel miss.

Nguồn: opencv

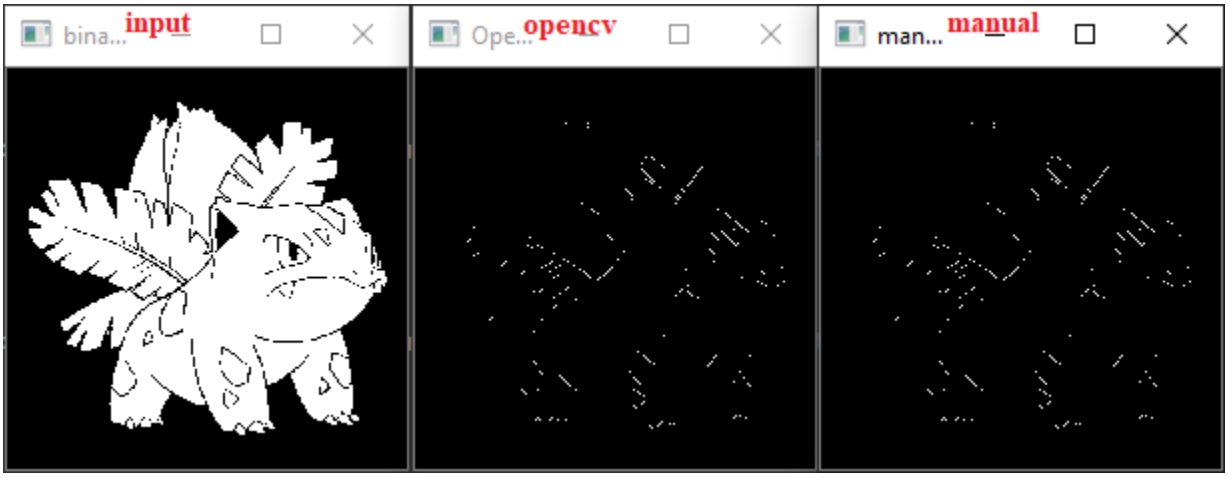
0	1	0
1	0	1
0	1	0

0	0	0
0	1	0
0	0	0

0	1	0
1	-1	1
0	1	0

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

Câu lệnh: main.py -i <input file> -o <output file> -p **hitmiss** -t <wait key time>



Nhận xét: Kết quả tương tự OpenCV.

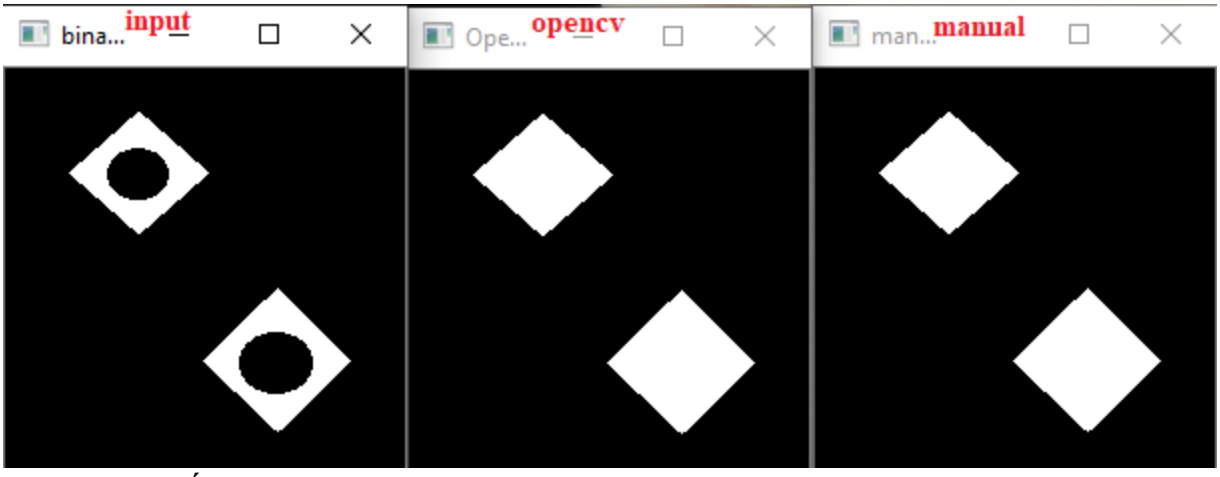
Boundary Extraction

Câu lệnh: main.py -i <input file> -o <output file> -p **boundary** -t <wait key time>

Nhận xét: Kết quả tương tự OpenCV.

Hole Filling

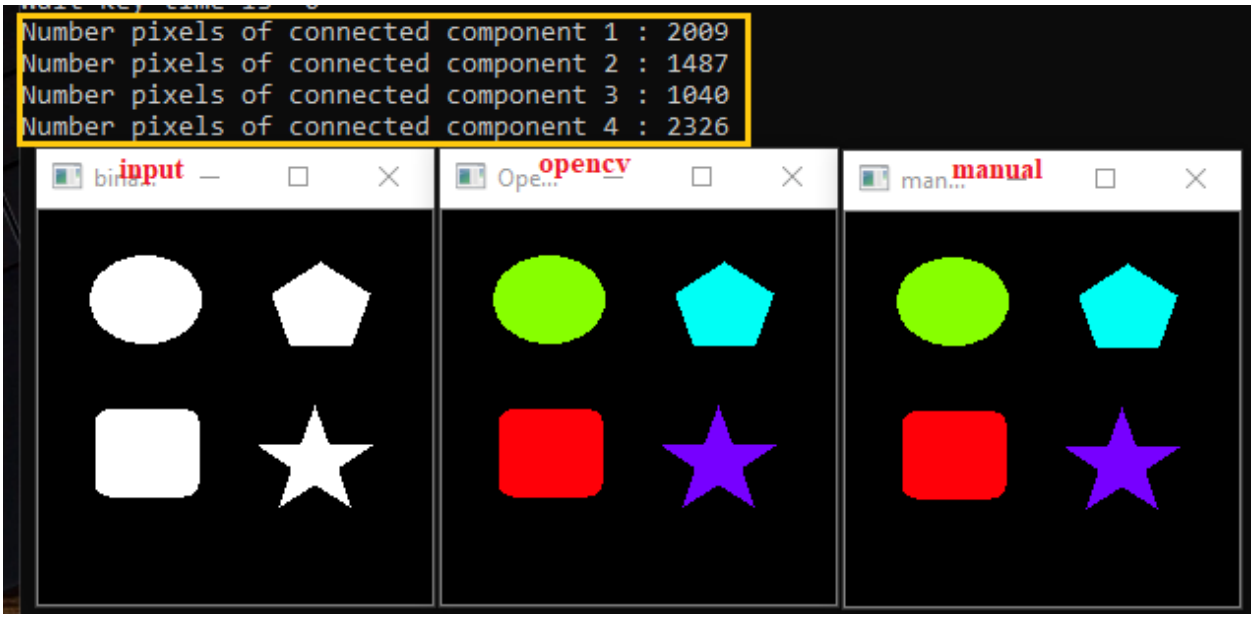
Câu lệnh: main.py -i <input file> -o <output file> -p **fillhole** -t <wait key time>



Nhận xét: Kết quả tương tự OpenCV.

Extraction of Connected Components

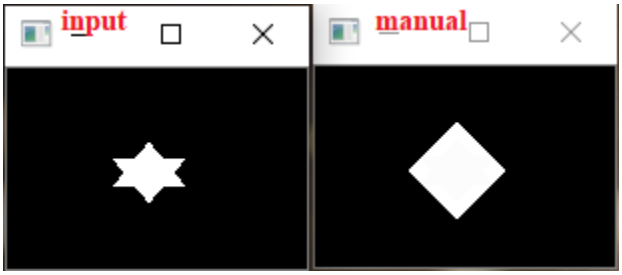
Câu lệnh: main.py -i <input file> -o <output file> -p **connectedcomponents** -t <wait key time>



Nhận xét: Kết quả trả về là ảnh đã gán nhãn cho các thành phần liên thông, số lượng thành phần liên thông và số pixel thuộc mỗi thành phần liên thông. Ảnh hiển thị sẽ tô các màu khác nhau cho các thành phần liên thông khác nhau. Kết quả hiển thị của hàm tự cài đặt tương tự kết quả hiển thị khi dùng hàm của opencv.

Convex Hull

Câu lệnh: main.py -i <input file> -o <output file> -p **convexhull** -t <wait key time>



Nhận xét: Kết quả chính xác tuy nhiên tốc độ xử lý chậm do nhiều lần lặp.

Thinning

Câu lệnh: main.py -i <input file> -o <output file> -p **thin** -t <wait key time>

Nhận xét: Kết quả gần giống OpenCV. Sự khác biệt là do hàm tự cài đặt không có bước chuyển đổi m-connectivity để loại bỏ multiple paths.

Thickening

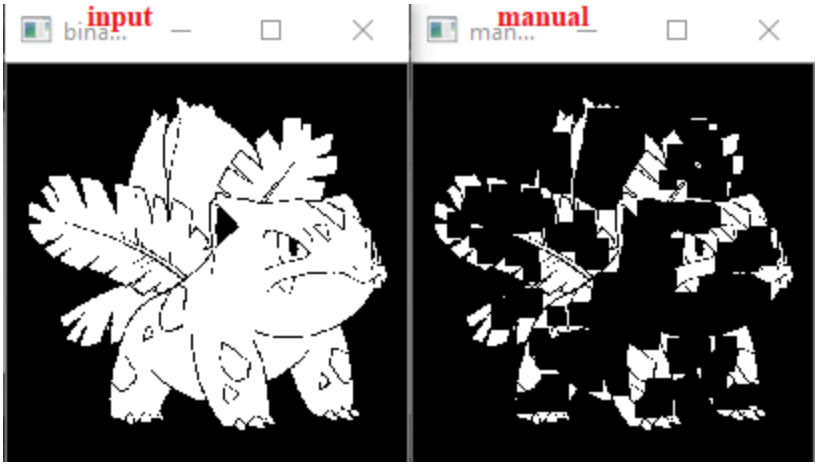
Câu lệnh: main.py -i <input file> -o <output file> -p **thicken** -t <wait key time>

Nhận xét: Nếu sử dụng cách complement input phối hợp với thinning thì foreground sẽ được làm dày đến nỗi gần như lắp đầy ảnh. Do đó hàm tự cài đặt sẽ sử dụng 8 kernels một lần duy nhất theo công thức:

$$A \odot \{B\} = ((\ldots((A \odot B^1) \odot B^2)\ldots) \odot B^n)$$

Skeleton

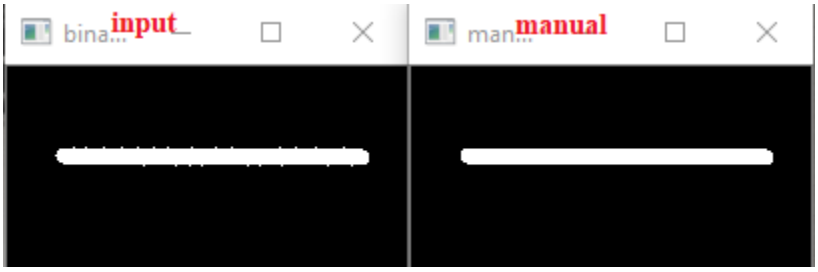
Câu lệnh: main.py -i <input file> -o <output file> -p **skeleton** -t <wait key time>



Nhận xét: Kết quả không được như mong đợi.

Pruning

Câu lệnh: main.py -i <input file> -o <output file> -p **prun** -t <wait key time>



Nhận xét: Hàm cài đặt cho kết quả tốt.

Mophological Reconstruction

Mã nguồn cài đặt bao gồm Reconstruction by Dilation, Reconstruction by Erosion, Opening by Reconstruction, Closing by Reconstruction.

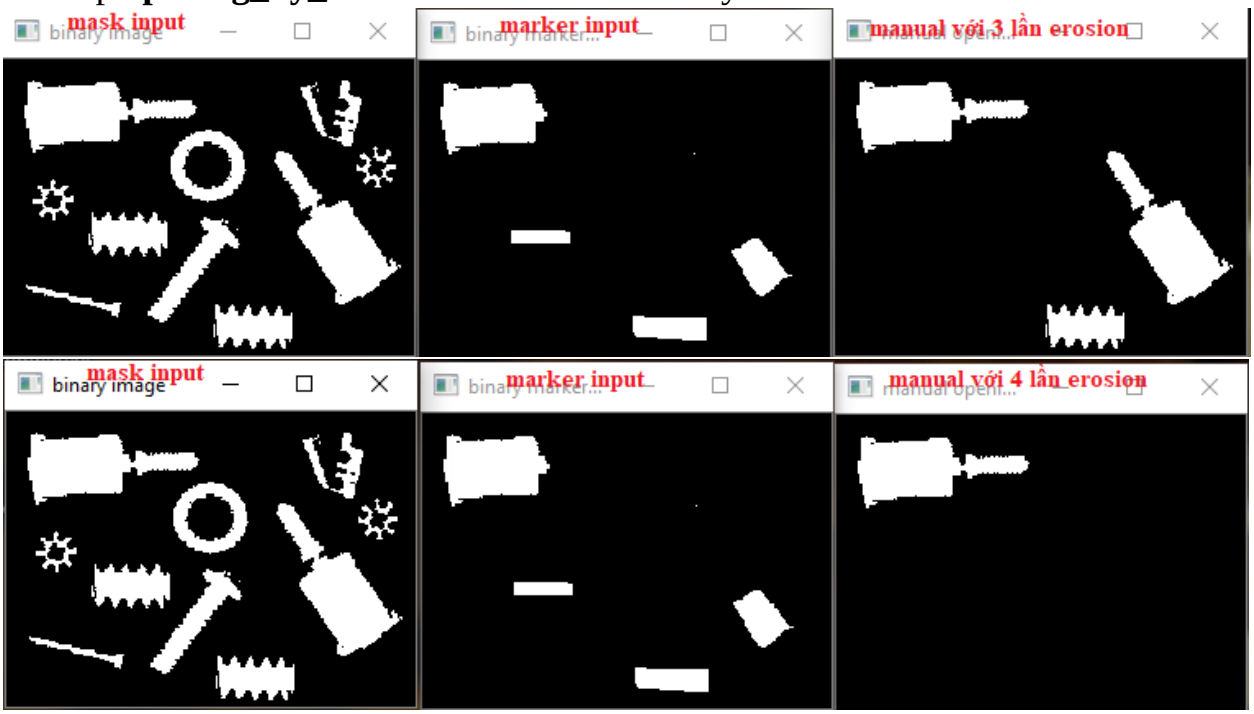
Demo cho phép Reconstruction by Dilation

Câu lệnh: main.py -i <mask file> -m <marker file> -o <output file> -p **reconstruct_by_dilation** -t <wait key time>

Nhận xét: Kết quả đã phục hồi được các đối tượng dựa vào marker.

Demo cho phép Opening by Reconstruction

Câu lệnh: main.py -i <mask file> -m <marker file> -k <số lần erosion> -o <output file> -p **opening_by_reconstruction**-t <wait key time>



Nhận xét: Phục hồi được các đối tượng dựa vào ảnh marker, có thể "lựa chọn" được đối tượng cần phục hồi dựa vào số lần erosion truyền vào.

2. Các toán tử hình thái học trên ảnh xám

Erosion

Câu lệnh:

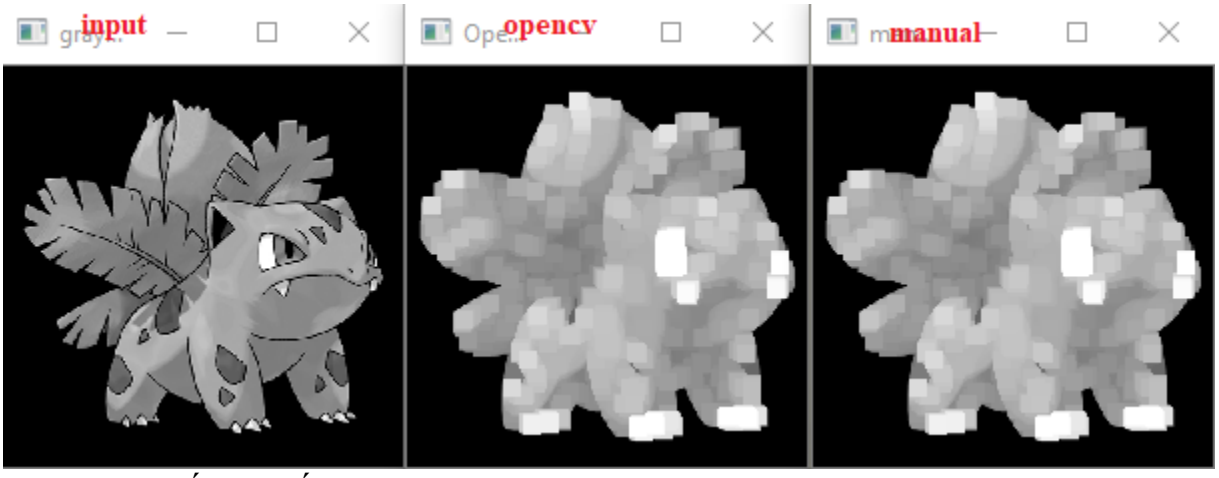
main.py -i <input file> -o <output file> -p **erode_gray** -t <wait key time>

Nhận xét: Kết quả tốt tương tự OpenCV.

Dilation

Câu lệnh:

main.py -i <input file> -o <output file> -p **dilate_gray** -t <wait key time>

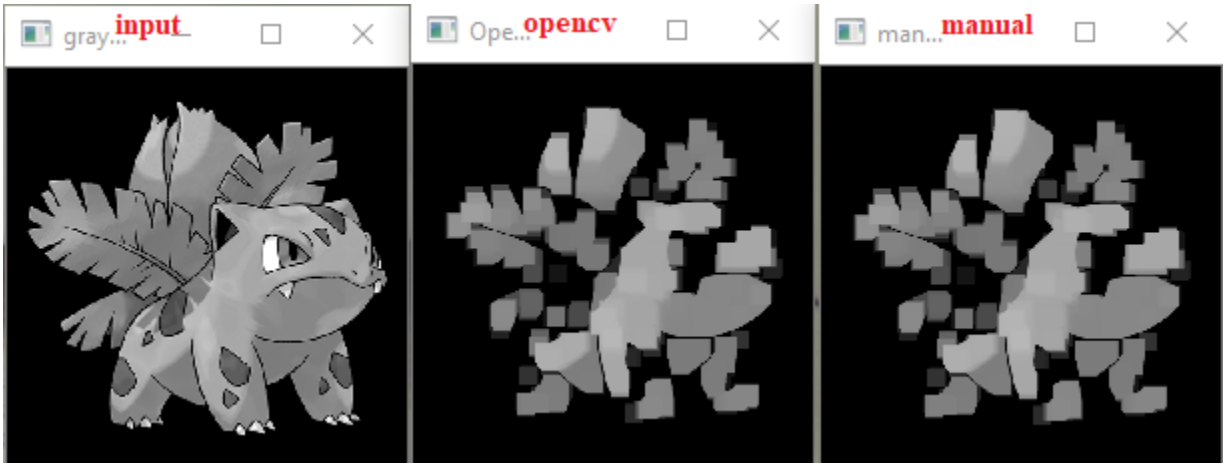


Nhận xét: Kết quả tốt tương tự OpenCV.

Opening

Câu lệnh:

main.py -i <input file> -o <output file> -p **open_gray** -t <wait key time>

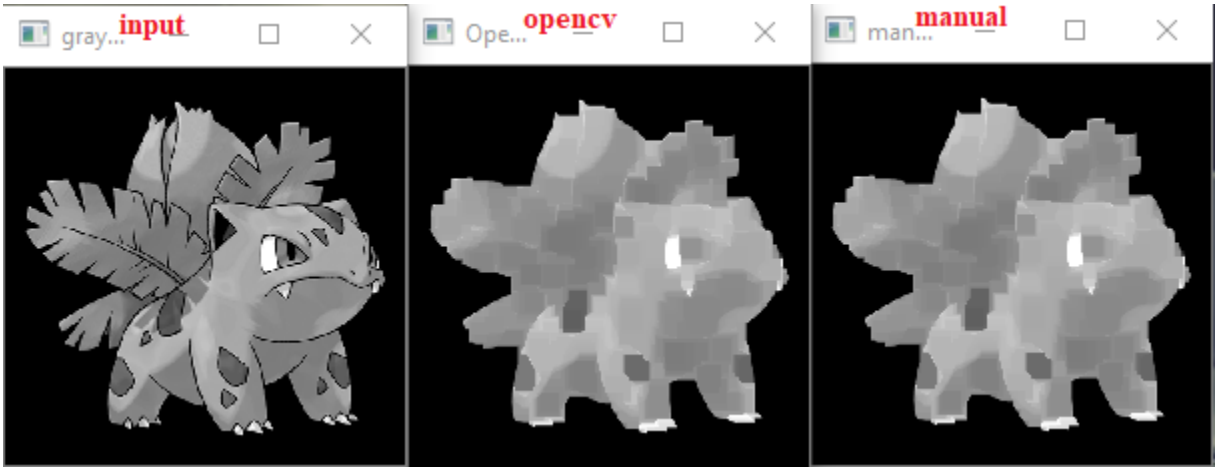


Nhận xét: Kết quả tốt tương tự OpenCV.

Closing

Câu lệnh:

main.py -i <input file> -o <output file> -p **close_gray** -t <wait key time>



Nhận xét: Kết quả tốt tương tự OpenCV.

Smoothing

Câu lệnh:

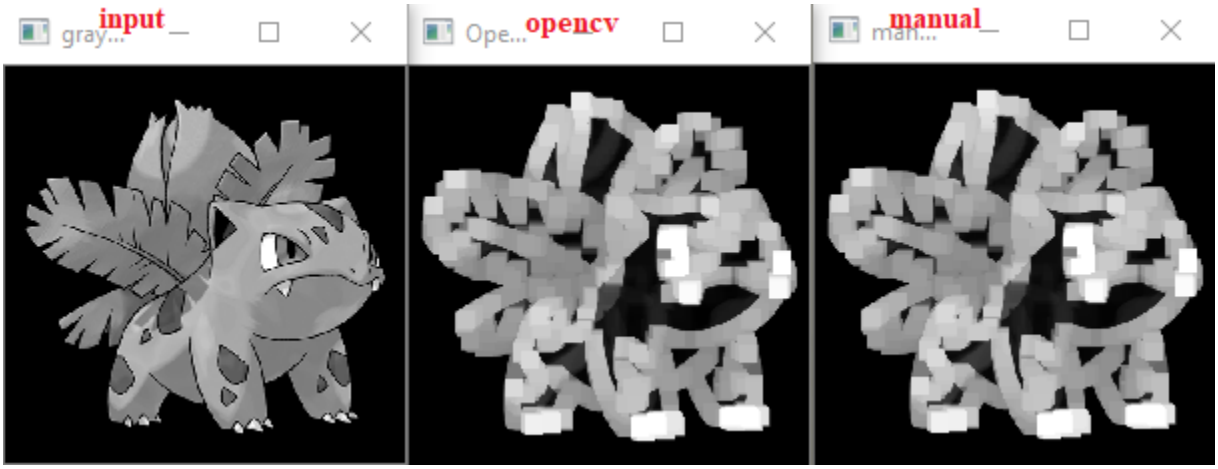
main.py -i <input file> -o <output file> -p **smooth_gray** -t <wait key time>

Nhận xét: Kết quả tốt tương tự OpenCV.

Gradient

Câu lệnh:

main.py -i <input file> -o <output file> -p **gradient_gray** -t <wait key time>

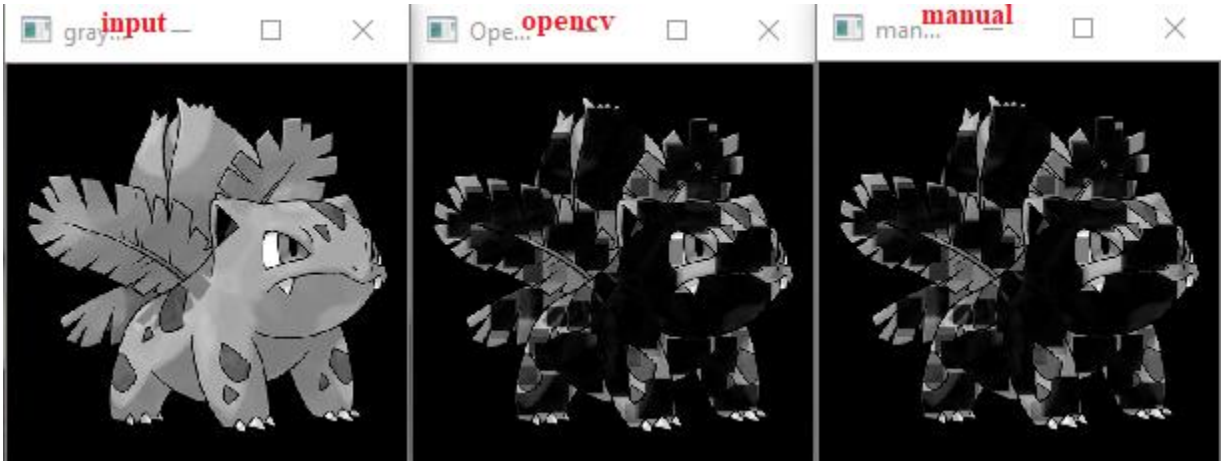


Nhận xét: Kết quả tốt tương tự OpenCV.

Top-hat transformation

Câu lệnh:

main.py -i <input file> -o <output file> -p **tophat_gray** -t <wait key time>

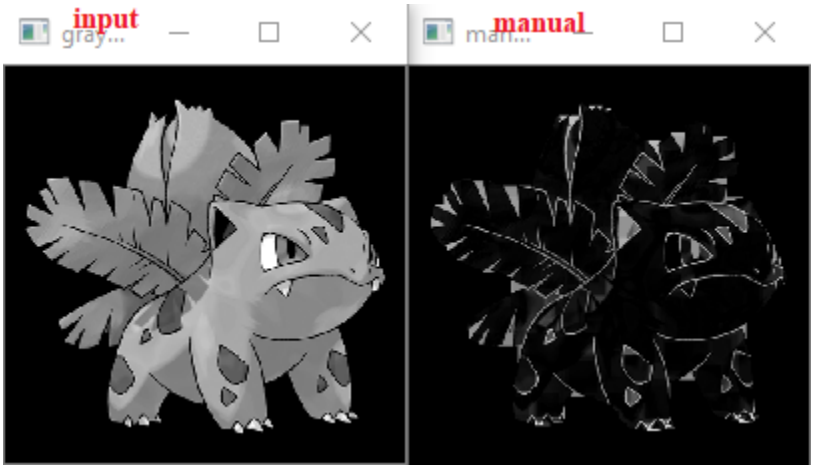


Nhận xét: Kết quả tốt tương tự OpenCV.

Bottom-hat Transformation

Câu lệnh:

main.py -i <input file> -o <output file> -p **bottom_gray** -t <wait key time>

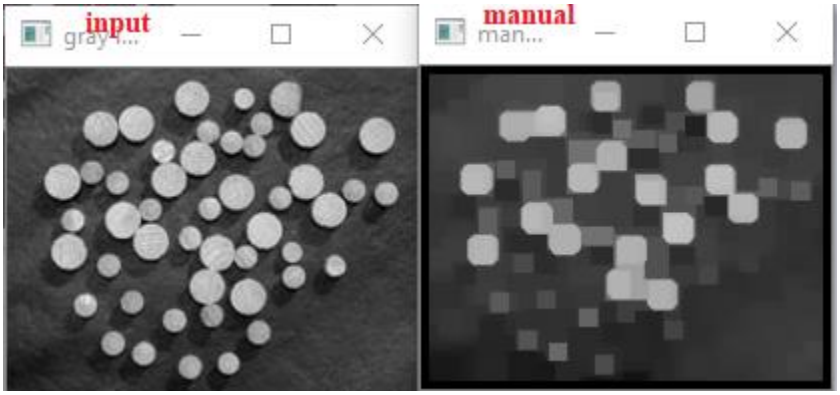


Nhận xét: Kết quả tốt.

Granulometry

Câu lệnh:

main.py -i <input file> -o <output file> -p **granulometry_gray** -t <wait key time>



Nhận xét: Kết quả như mong đợi.

Mophological Reconstruction

Mã nguồn cài đặt bao gồm Reconstruction by Dilation, Reconstruction by Erosion, Opening by Reconstruction, Closing by Reconstruction.

Demo cho phép Reconstruction by Dilation

Câu lệnh:

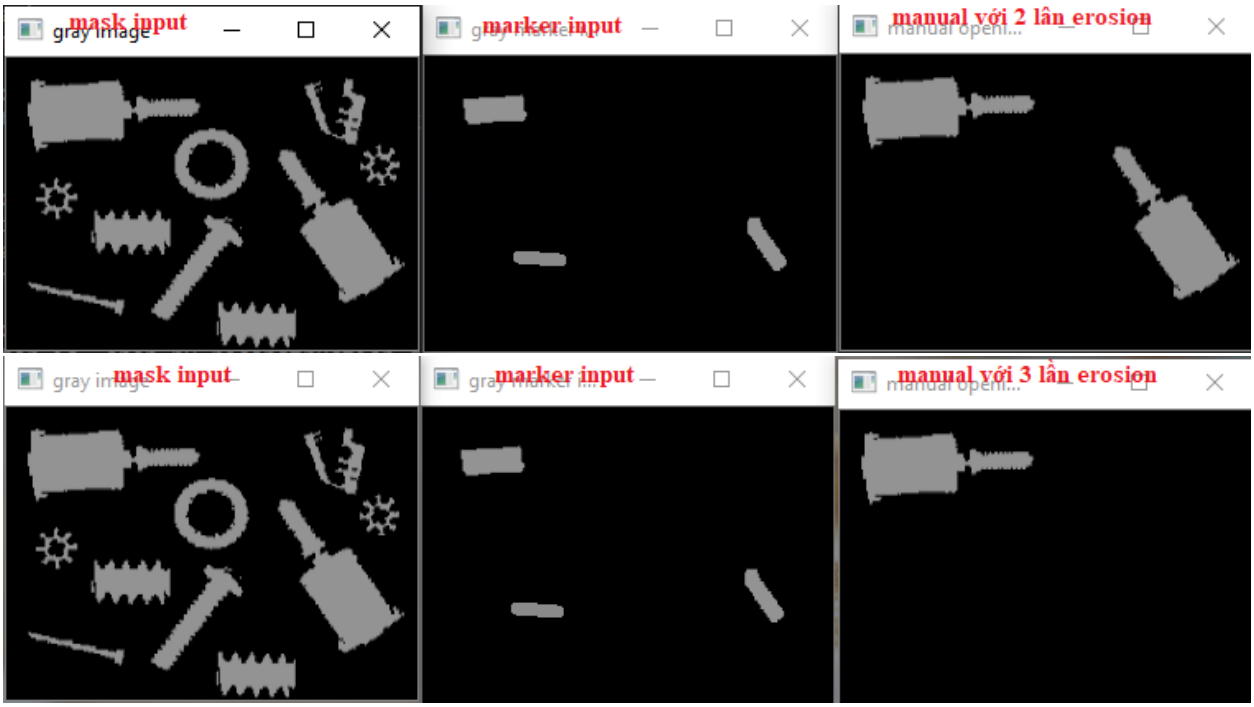
main.py -i <mask file> -m <marker file> -o <output file> -p **reconstruct_by_dilation_gray** -t <wait key time>

Nhận xét: Kết quả đã phục hồi được các đối tượng dựa vào marker.

Demo cho phép Opening by Reconstruction

Câu lệnh:

main.py -i <mask file> -m <marker file> -k <số lần erosion> -o <output file> -p **open_by_reconstruction_gray** -t <wait key time>



Nhận xét: Phục hồi được các đối tượng dựa vào ảnh marker, có thể "lựa chọn" được đối tượng cần phục hồi dựa vào số lần erosion truyền vào.

Hết