HƯỚNG DẪN THỰC HÀNH #01

CÁC TOÁN TỬ HÌNH THÁI HỌC

(*Keyword: Morphological Operations)*

I. Mục tiêu

- Học viên tự cài đặt được các toán tử hình thái học và so sánh với hàm thư viện.

II. Yêu cầu cài đặt

- Ngôn ngữ lập trình: *Python*, phiên bản tối thiểu khuyến nghị **3.6**.
- Thư viện: *NumPy, OpenCV-Python,* (có thể tham khảo thêm *OpenCV_Contrib* và *Scipy*).
- IDE / Text Editor: khuyến nghị sử dụng *JetBrains PyCharm Community* (*PyCharm*) hoặc *Microsoft Visual Studio Code (VS Code)*.

III. Yêu cầu bài nộp

1. Cấu trúc thư mục
   - doc: các tệp báo cáo bao gồm cả file *MSHV_report_p01.doc/docx/latex* và *MSHV_report_p01.pdf*
   - source: chứa toàn bộ mã nguồn, đã loại bỏ các file tạm, trung gian, các tệp đã biên dịch nếu có...
   - bonus: tùy chọn, cho điểm cộng, nếu có

   Học viên vẫn phải nộp phần báo cáo riêng như trên, kể cả trường hợp sử dụng Jupyter Notebook (đặt tên là *MSHV_p01*).
2. Các yêu cầu khác
   - Báo cáo trình bày rõ ràng, trực quan: có bảng tự đánh giá kết quả của công việc so với các nội dung yêu cầu tương ứng (0-100%)
   - Danh sách các chức năng có trong chương trình với hình ảnh chứng minh, tóm tắt cách sử dụng và cài đặt (ví dụ: có thể thông qua mã giả, mô tả các phương pháp hoặc cách thực hiện, không sao chép mã nguồn vào báo cáo).
   - Mã nguồn cần comment ở các dòng tương ứng.

   Chú ý: Nghiêm cấm sao chép bài của học viên khác, nếu bị phát hiện, cả người sao chép và người bị sao chép đều sẽ bị xem xét xử lý.

IV. Nội dung

1. Cài đặt các thành phần cần thiết:
   - Python: https://www.python.org/

- Các thư viện:

+ Khuyến nghị tạo môi trường ảo riêng, tham khảo cho *PyCharm* và *VS Code* tại
https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html
https://code.visualstudio.com/docs/python/environments

+ Khuyến nghị cài đặt các thư viện bằng *Pip*, tham khảo tại
https://www.w3schools.com/python/python_pip.asp
https://packaging.python.org/tutorials/installing-packages/

2. Cài đặt các toán tử hình thái học không sử dụng hàm xây dựng sẵn của các thư viện, có thể tham khảo các hàm xây dựng sẵn trong phần mã nguồn tham khảo tương ứng để so sánh, kiểm tra kết quả...

Tối thiểu: các toán tử đã học, các toán tử được học trên lớp lý thuyết.

Mở rộng: các toán tử khác trong tài liệu được cung cấp ở lớp lý thuyết…

2.1. Toán tử hình thái học nhị phân

Sử dụng thư viện *OpenCV* và *NumPy* để đọc và tiến hành các thao tác xử lý ảnh nhị phân.

```
# import required libraries
# import OpenCV2
import cv2
# import NumPy
import numpy as np


"""
Using cv2.imread to read image file, 0 flag means grayscale (in
this case, input.png is a binary image, so we do not care about
color channels)
"""
img_gray = cv2.imread('input.png', 0)

# convert to binary image by using auto-calculate threshold
(thresh, img) = cv2.threshold(img_gray, 128, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)
# or using predetermined threshold
thresh = 127
im_bw = cv2.threshold(img_gray, thresh, 255, cv2.THRESH_BINARY)

# Using cv2.imshow to display the loaded image
cv2.imshow('Binary image', img)
```

- Toán tử giãn nở nhị phân (*Binary Dilation*)

```
# define kernel for using in binary dilation
kernel = np.ones((3,3),np.uint8)
# or
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3,3))

# Using cv2.erode to apply binary dilation, with defined kernel
and only one iteration (this param could be skipped)
img_dilation = cv2.dilate(img, kernel, iterations=1)
```

- Toán tử co nhị phân (*Binary Erosion*)

```
img_erosion = cv2.erode(img, kernel)
```

- Toán tử mở nhị phân (*Binary Opening*)

```
img_opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

- Toán tử đóng nhị phân (*Binary Closing*)

```
img_closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

- Toán tử Hit-or-Miss

```
img_hit_or_miss = cv2.morphologyEx(img, cv2.MORPH_HITMISS,
kernel)
```

- Thinning

```
# this function available in opencv-contrib-python
(OpenCV_Contrib)
img_thinning = cv2.ximgproc.thinning(img)
```

Từ đó, mở rộng hơn với các biến đổi như: *Boundary Extraction*, *Hole Filling*, *Extraction of Connected Components*, *Convex Hull*, *Thickening*, *Skeletons*, *Pruning*, *Morphological Reconstruction*.

2.2. Toán tử hình thái học với ảnh độ xám

Sử dụng thư viện *OpenCV* và *NumPy* để đọc và tiến hành các thao tác xử lý ảnh độ xám.

```
# read image directly then convert to img_gray using
appropriate flag
img_origin = cv2.imread('input.png')
img = cv2.cvtColor(img_origin, cv2.COLOR_BGR2GRAY)
```

Tương tự với toán tử hình thái học nhị phân, cài đặt các toán tử hình thái học với ảnh độ xám như: giãn nở (*Dilation*), co (*Erosion*), mở (*Opening*), đóng (*Closing*), *Morphological Reconstruction* với ảnh độ xám Gray-Scale.

Bổ sung thêm các toán tử như:

- Morphological Gradient (*dilation - erosion*)

```
img_gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT,
kernel)
```

- Biến đổi Top-Hat

```
img_tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
```

- Biến đổi Black-Hat

```
img_blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT,
kernel)
```

Và mở rộng hơn với các nội dung như: *Granulometry*, *Textural Segmentation*.

3. Tham khảo phần cài đặt mẫu cho toán tử giãn nở nhị phân (*Binary Dilation*), chú ý tham khảo về cấu trúc mã nguồn cũng như tham số dòng lệnh và các phần ghi chú *TODO* cần thực hiện.