HW7

Date: 2021. 11. 3
Student ID: 21600635
Name: Bomoon JUNG

1. Homework 7

1) Develop a program which performs various kinds of thresholding

- ✓ Use 'finger_print.png', 'adaptive_1.jpg', 'adaptive.png'.
- ✓ Read all images as gray scale images.
- ✓ Select and apply a threshold method for each image.
  - For 'finger_print.png', set fingerprint region to 0 and background region to 255.
  - For 'adaptive_1.jpg', and 'adaptive.png', set character region to 0 and background region to 255.
- ✓ Display results on three windows
  - Windows name: 'finger_print', 'adaptive_1', and 'adaptive'
- ✓ In your report, include explanations of why, for each image, you used those algorithms, argument values, and so forth.

2) explanation

1. finger_print.png: In this picture, it is easy to distinguish between the background and the object. Therefore, thresholding can be performed easily with one threshold. Therefore, the Basic method of Global Thresholding was used. Basic method sets thresh_T first. And after making two groups using thresh_T, find the average of each group. Then, this step is repeated until a suitable thresh_T is reached. Finally, input and output are assigned to the threshold function, and set as THRESH_BINARY.

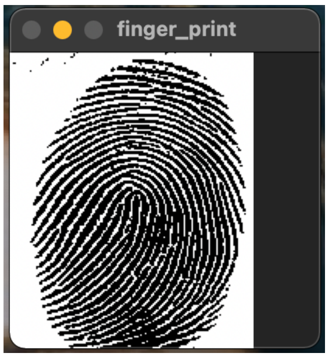
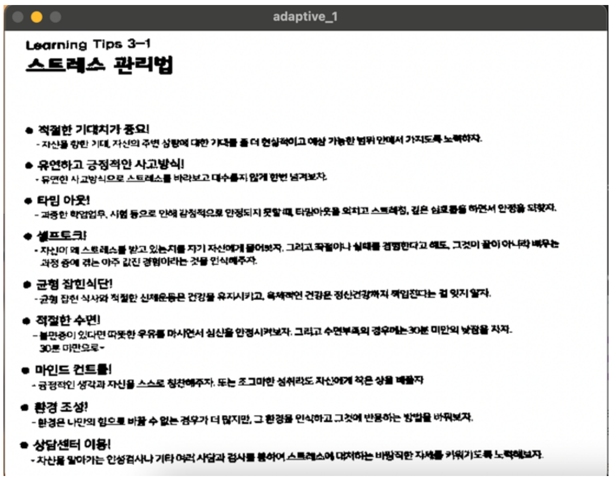


Figure 1. result of finger_print

2. adaptive_1: If you look at the bottom right of this picture, you can see that it is a little dark. Therefore, using local thresholding, the threshold is determined by considering adjacent pixels. Therefore, much better thresholding can be implemented. Therefore, I used the adaptiveThreshold function, and input image, output image, max value, method, threshold type, block size of an appropriate size, and constant are inserted in order from the first argument. At this time, the threshold type is set to 'THRESH_BINARY', so the pixel value is set as the max value for the background that does not exceed the threshold.



Learning Tips 3-1
스트레스 관리법

● 적절한 기대치가 중요!
- 자신을 향한 기대, 자신의 주변 상황에 대한 기대를 좀 더 현실적이고 예상 가능한 범위 안에서 가지도록 노력하자.

● 유연하고 긍정적인 사고방식!
- 유연한 사고방식으로 스트레스를 바라보고 대수롭지 않게 한번 넘겨보자.

● 타임 아웃!
- 과중한 학업업무, 시험 등으로 인해 감정적으로 안정되지 못할 때, 타임아웃을 외치고 스트레칭, 깊은 심호흡을 하면서 안정을 되찾자.

● 셀프토크!
- 자신이 왜 스트레스를 받고 있는지를 자기 자신에게 물어보자. 그리고 좌절이나 실패를 경험한다고 해도, 그것이 끝이 아니라 배우는 과정 중에 겪는 아주 값진 경험이라는 것을 인식해주자.

● 균형 잡힌 식단!
- 균형 잡힌 식사와 적절한 신체운동은 건강을 유지시키고, 육체적인 건강은 정신건강까지 책임진다는 걸 잊지 말자.

● 적절한 수면!
- 불면증이 있다면 따뜻한 우유를 마시면서 심신을 안정시켜보자. 그리고 수면부족의 경우에는 30분 미만의 낮잠을 자자. 30분 미만으로~

● 마인드 컨트롤!
- 긍정적인 생각과 자신을 스스로 칭찬해주자. 또는 조그마한 성취라도 자신에게 작은 상을 베풀자

● 환경 조성!
- 환경은 나만의 힘으로 바꿀 수 없는 경우가 더 많지만, 그 환경을 인식하고 그것에 반응하는 방법을 바꿔보자.

● 상담센터 이용!
- 자신을 알아가는 인성검사나 기타 여러 사담과 검사를 통하여 스트레스에 대처하는 바람직한 자세를 키워가도록 노력해보자.

Figure 2. result of adaptive_1

3. adaptive: Banding noise is present in this figure. Therefore, local thresholding should be used for each column. In that case, much better thresholding can be implemented because the threshold is determined by considering the intensity for each column. Therefore, the adaptiveThreshold function is used in the same way as 'adaptive_1', but the difference is that thresholding is performed for each column of the image. And by inserting the block size as much as the number of rows, thresholding is taken precisely in column unit. Therefore, banding noise can be solved.

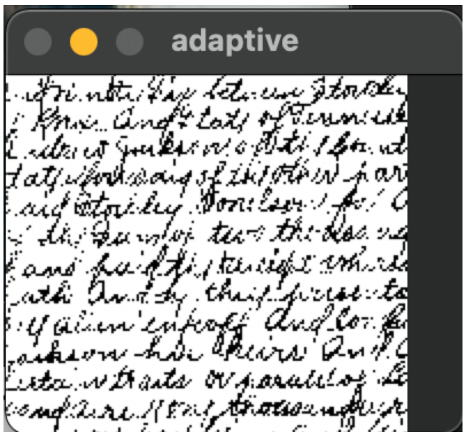


Figure 3. result of adaptive

3) Source code

```cpp
// g++ $(pkg-config --cflags --libs opencv) HW7_21600635.cpp -o ./a

#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>

using namespace cv ;
using namespace std ;

Mat
basic (Mat src) {
    int thresh_T = 200 ;
    int th = 10 ;
    int low_cnt = 0 ;
    int high_cnt = 0 ;
    int low_sum = 0 ;
    int high_sum = 0 ;

    Mat image = imread("./resources/Lena.png", 0) ;

    while(1) {
        for( int j = 0 ; j < src.rows ; j++ ) {
            for( int i = 0 ; i < src.cols ; i++ ) {
                if( src.at<uchar>(j, i) < thresh_T) {
                    low_sum += src.at<uchar>(j, i) ;
                    low_cnt++ ;
                }
                else {
                    high_sum += src.at<uchar>(j, i) ;
                    high_cnt++ ;
                }
            }
        }
        if( abs(thresh_T - (low_sum / low_cnt + high_sum / high_cnt) / 2.0f) < th)
{
            break ;
        }
        else {
            thresh_T = (low_sum / low_cnt + high_sum / high_cnt) / 2.0f ;
            low_cnt = high_cnt = low_sum = high_sum = 0 ;
        }
    }

    Mat thresh ;
    threshold(src, thresh, thresh_T, 255, THRESH_BINARY) ;

    return thresh ;
}
```

```
Mat
local_adaptive (Mat src) {
    Mat adaptive_binary ;
    adaptiveThreshold(src, adaptive_binary, 255, ADAPTIVE_THRESH_MEAN_C,
THRESH_BINARY, 85, 15) ;

    return adaptive_binary ;
}

Mat
local_adaptive_partition (Mat src) {
    int mask = src.rows ;
    for ( int i = 0 ; i < src.cols ; i ++ ) {
        Rect rect(i, 0, 1, src.rows) ;
        adaptiveThreshold(src(rect), src(rect), 255, ADAPTIVE_THRESH_MEAN_C,
THRESH_BINARY, mask, 15) ;
    }

    return src ;
}

int
main() {
    Mat finger_print = imread("HW7_resources/finger_print.png", 0) ;
    Mat adaptive_1 = imread("HW7_resources/adaptive_1.jpg", 0) ;
    Mat adaptive = imread("HW7_resources/adaptive.png", 0) ;

    if ( finger_print.empty() |  adaptive_1.empty() | adaptive.empty() ) {
        cout << "no such file" << endl ;
        return 0 ;
    }

    finger_print = basic(finger_print) ;
    adaptive_1 = local_adaptive(adaptive_1) ;
    adaptive = local_adaptive_partition(adaptive) ;

    imshow("finger_print", finger_print) ;
    imshow("adaptive_1", adaptive_1) ;
    imshow("adaptive", adaptive) ;
    waitKey(0) ;
    return 0 ;
}
```