HW8

Date: 2021. 11. 16
Student ID: 21600635
Name: Bomoon JUNG

1. Homework 8

1) Develop a program which performs background substraction.

- ✓ Read ‘background.mp4’ as gray scale
- ✓ Set background image as the average of the first 10 frames.
- ✓ Generate a binary image by using the following equation.
  - Result(x, y) = $\begin{cases} 255 \mid Current\ frame(x,y) - Background(x,y) \mid > 20 \\ 0 \quad otherwise \end{cases}$
- ✓ Draw bounding rectangle on the original input video each moving object whose size is bigger than 400 pixels.
- ✓ Print out the number of moving objects on the image whose size is bigger than 400 pixels.
- ✓ Display three windows.
  - A window showing the background image
  - A window showing the Result (x, y) video
  - A window showing the final result video (with rectangles and text)

2) explanation

I. function description

- background_img: This is a function that takes an image as an argument and extracts the average over the entire gray scale image time as a background.

- draw_rect: It takes 1 frame of the image and the extracted object as arguments, and if the size of the object detected through ‘boundingRect’ exceeds 400 pixels, a box is drawn using a rectangle.

- main: After reading the ‘background.mp4’ video, the background is obtained through the background_img function. Then, after obtaining a 'foregroundMask' for each frame, it makes image processing easier by using a threshold. Finally, after using the draw_rect function to draw a box on objects of a certain size, the foregroundMask and the resulting value are output.

3) Source code

```cpp
// g++ $(pkg-config --cflags --libs opencv) HW8_21600635.cpp -o ./a

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

using namespace cv ;
using namespace std ;

Mat
background_img ( VideoCapture src ) {
    Mat background ;
    Mat image ;
    Mat gray ;
    Mat avg ;
    src >> avg ;
    cvtColor( avg, avg, CV_RGB2GRAY ) ;

    int cnt = 2 ;

    while ( true ) {
        if ( !src.read( image ) )
            break ;
```

```
        cvtColor( image, gray, CV_BGR2GRAY ) ;
        add(gray / cnt, avg * (cnt - 1) / cnt, avg) ;
        cnt++ ;

        int current_frame = src.get( CAP_PROP_POS_FRAMES ) ;
    }

    return avg ;
}

Mat
draw_rect ( Mat src, Mat foregroundMask ) {
    vector<vector<Point> > contours ;
    vector<Vec4i>hierarchy ;
    findContours( foregroundMask, contours, hierarchy, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_SIMPLE ) ;


    // defining bounding rectangle
    vector<Rect> boundRect(contours.size()) ;
    for ( int i = 0 ; i < contours.size() ; i++ ) {
        boundRect[i] = boundingRect(Mat(contours[i])) ;
    }
    // draw rectangles on the contours
    int count = 0 ;
    for ( int i = 0 ; i < contours.size() ; i++ ) {
        if (boundRect[i].width * boundRect[i].height > 400) {
            count++ ;
            rectangle( src, boundRect[i].tl(), boundRect[i].br(), Scalar(255, 0,
0), 2, 8 ,0) ;
        }
    }
    putText(src, format("object: %d", count), Point(5, 20), 1, 0.8, Scalar(255,
255, 255), 1, 8) ;



    return src ;
}

int
main () {
    VideoCapture capture ;
    VideoCapture capture_for_background ;
    // open mp4 file
    if ( capture.open( "background.mp4") == 0 ||
capture_for_background.open("background.mp4") == 0 ) {
        cout << "no such file" << endl ;
        return 0 ;
    }
```

```
    Mat background = background_img( capture_for_background ) ;
    Mat gray ;
    Mat image ;
    Mat foregroundMask ;
    imshow( "background", background ) ;

    cout << "pass" << endl ;
    while ( true ) {

        if ( capture.grab() == 0 )
            break ;

        capture.retrieve( image ) ;
        cvtColor( image, gray, CV_BGR2GRAY ) ;

        absdiff( background, gray, foregroundMask ) ;
        threshold( foregroundMask, foregroundMask, 20, 255, CV_THRESH_BINARY ) ;
        gray = draw_rect ( gray, foregroundMask ) ;
        imshow( "Result(x, y)", foregroundMask ) ;
        imshow( "result", gray) ;


        waitKey( 33 ) ;
    }
    return 0 ;
}
```