LAB 1. Color Image Segmentation

Date: 2021.04.17
Student ID:
21500482, 21600144
Name: 이두용, 김주호

1. Introduction

This experiment deals the image segmentation method which is used to detect and measure the maximum and average temperature of the mask-worn faces. There are two parts for this lab: part 1 is to draw contour and rectangle on the area where temperature is to be measured, and the second part is to convert the color data into intensity value which is then converted into temperature magnitude. Various methods, too, are adopted including morphology, filtering, and finding contours. The lab progress algorithm flow chart is in **Figure 1.**

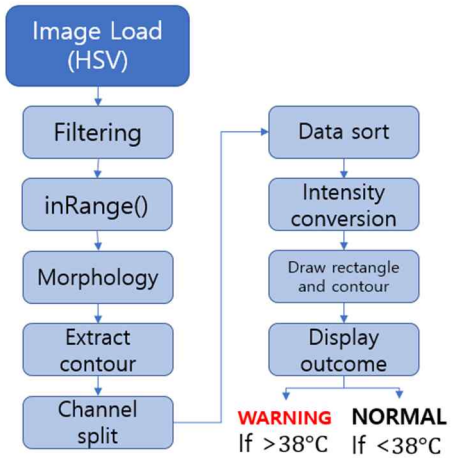


Figure 1. Flow chart of the lab

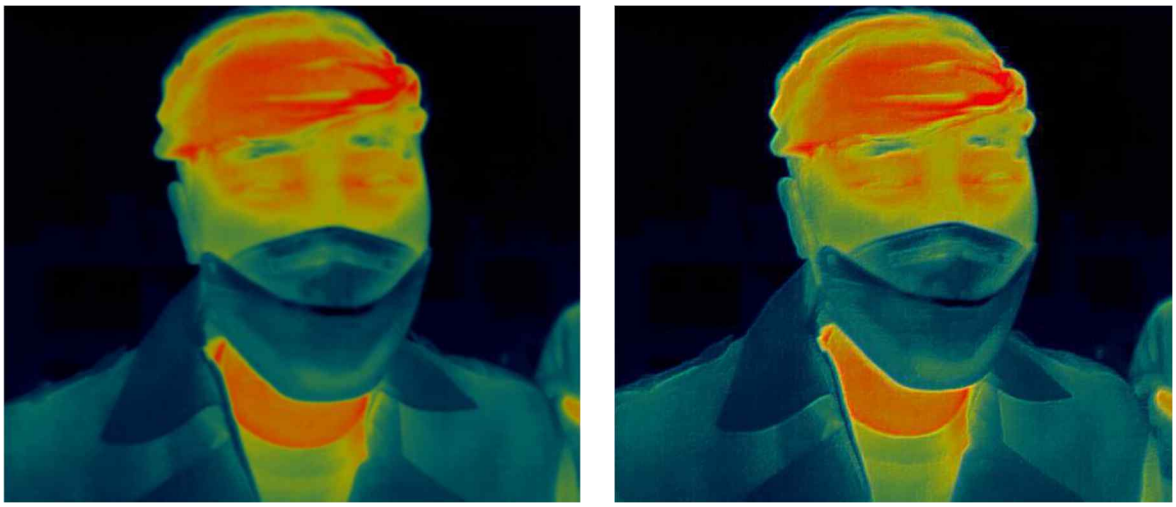
2. Procedure

Part 1. Face Segmentation excluding mask.

The target video used for this experiment is shown in **Figure 1.**

Figure 1. The captured image of the target video.

Since half of the face is covered by a mask, we only need to consider the exposed skin, including face and neck that are not covered by either cloths or a mask. The first step is to import the video and convert the color space from RGB to HSV. Gaussianblur() and blur() filters are not quite adequate in this case because they may deteriorate the color composition. Sobel filter was adopted after the video was loaded to enhance edge detection and maximize color data acquisition. The filtered image comparison is depicted in **Figure 2.**



a) Gaussian Blur b) Sobel

Figure 2. Filtered image comparison.

It is obvious that the color edges are clearer when the Sobel filter is used. Using the inRange() function, the ranges of each hue, saturation, and value magnitudes are determined, in order to segment out desired color information. The minimum and maximum magnitudes corresponding to the skin temperature color are defined using track bars where h is 0~50, s is 50~255, and v is 150~255. Then, morphology was used to make the extracted result from the previous step to be more apparent. **Figure 3** demonstrates how different types of morphology can affect the image after thresholding.

a) inRange() b) Erosion c) Dilation d) Open e) Close

Figure 3. Post-process results using morphology.

From the images above, either dilation or close method seems promising for further image processing. However, the closed method removes unnecessary pixels existing outside the region of interest, so the closed morphology was used. The subsequent step is to detect contours of the connected objects. One important consideration is selecting the proper contour in the face. This was managed by using the contour size function where only the contour sizes having larger than 17 were drawn. Additionally, rectangular box outlines identifying the contours were drawn as well. The result of part 1 is shown in **Figure 4.**

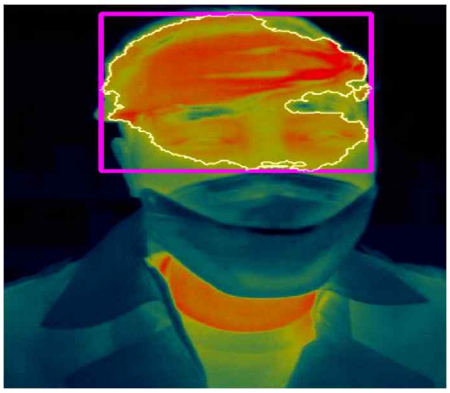


Figure 4. Result video image after drawing contour and rectangle.

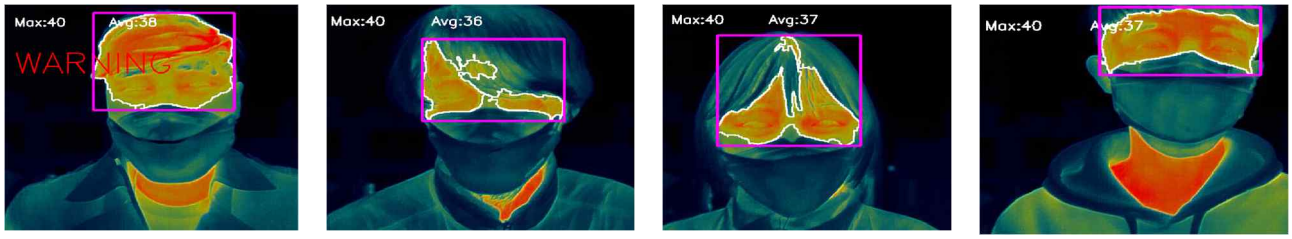
Part 2. Temperature Measurement

In this part of the lab, we are going to analyze the temperature intensity data of the image scaled within the pre-defined temperature range, $25{\sim}40°C$. In order to manipulate the color data into gray temperature intensity, several steps need to be processed. In our case, we used the split() function to divide the image color HSV components into three individual arrays. Only the value channel of HSV color space was used for intensity computation. Successively, the value array was rearranged from two dimension to one dimension array to sort in a descending manner – largest to smallest. The average temperature corresponding to the region of interest will not be valid because the room temperature is relatively lower than the facial temperature. Not every single dataset is quite necessary for temperature calculation. To be more specific, if there exists an object with higher temperature than a face, a cup of coffee for instance, the average or maximum temperature result can be distorted. Therefore, only the top 3~5% of data was attained, removing all the room temperature intensity data, and first 0.5% data was eliminated just in case of an unexpected hot

element attained by the sensor. By using a simple equation below, the intensity was converted into temperature ranging from $25{\sim}40°C$.

$$T = 25 + \frac{I_v}{255} \cdot 15$$

After temperature conversion, the average temperature was obtained by adding all the acquired data's magnitude and dividing the number of the data, and the maximum temperature indicates the first index of the data array. Additionally, the warning message must display if the average temperature exceeds $38°C$. Printing text messages is done using the putText() function. The final outcome of this lab is shown in **Figure 5.**



a) 1st person b) 2nd person c) 3rd person d) 4th person

Figure 5. Contour, rectangle, average/maximum temperature indication result.

3. Conclusion

This lab mainly dealt with conversion of an IR image to HSV color space, thresholding each H,S, and V channel with specified minimum and maximum value of the interested region, and utilizing the intensity data in the value channel. The importance of selecting proper filter type and morphology method was once again emphasized. We learned new image processing techniques such as obtaining the contour and intensity data for the region of interest and sorting the data to process the given requirements.

4. Appendix

```
//#include "opencv2/video/tracking.hpp"
//#include "opencv2/imgproc/imgproc.hpp"
//#include "opencv2/highgui/highgui.hpp"
//#include <ctype.h>
#include <iostream>
#include <opencv2/opencv.hpp>

using namespace cv;
using namespace std;

#define OUTPUT_VIDEO_NAME "Result.avi"

int main()
{
    Mat image, image_disp, hsv, dst, dst_filter, dst_morph, mul;
    vector<vector<Point> > contours;
```

```
vector<Point2f>center(contours.size());
vector<float>radius(contours.size());
vector<Vec4i> hierarchy;
VideoCapture cap("IR_DEMO_cut.AVI");
VideoWriter videoWriter;



if (!cap.isOpened())   // if not success, exit the programm
{
    cout << "Cannot open the video cam₩n";
    return -1;
}
int key = 0;

//write video
float videoFPS = cap.get(cv::CAP_PROP_FPS);
int videoWidth = cap.get(cv::CAP_PROP_FRAME_WIDTH); //650
int videoHeight = cap.get(cv::CAP_PROP_FRAME_HEIGHT); //512
videoWriter.open(OUTPUT_VIDEO_NAME, VideoWriter::fourcc('M', 'J', 'P', 'G'),
videoFPS, Size(videoWidth, videoHeight), true);

while (1)
{
    bool bSuccess = cap.read(image);
    Mat dst_track = Mat::zeros(image.size(), CV_8UC3);
    image.copyTo(image_disp);
    if (!bSuccess)   // if not success, break loop
    {
        cout << "Cannot find a frame from  video stream₩n";
        break;
    }
    key = waitKey(30);
    if (key == 27) // wait for 'ESC' press for 30ms. If 'ESC' is pressed, break loop
    {
        cout << "ESC key is pressed by user₩n";
        break;
    }

    for (;;)
    {
        cap.read(image);
        imshow("Source", image);
        Mat grad_x, grad_y,filtered;

        Sobel(image, grad_x, CV_8U, 1, 0);
        Sobel(image, grad_y, CV_8U, 0, 1);
        addWeighted(grad_x, 0.3, grad_y, 0.3, 0,filtered);

        image = image + filtered ;
        imshow("sobel", image);
        //GaussianBlur(image, image, Size(3, 3), 3);
        //imshow("gaussian", image);

        cvtColor(image, hsv, COLOR_BGR2HSV);  // HSV로 변환
```

```
            image.copyTo(image_disp);

            //========================== filtering=================================/
            //GaussianBlur(hsv, dst_filter, Size(3, 3), 1);
            //imshow("filter", dst_filter);
            //=========================================================== ============/

            //-------------------- set dst as the output of InRange---------------------/
            inRange(hsv, Scalar(0, 100, 150), Scalar(50, 255, 255), dst); // inRange를 통해
범위 제한
            //namedWindow("InRange", 0);
            imshow("InRange", dst);
            //------------------------------------------------------------------------- /

            //============================morphology=================================/
            Mat element = getStructuringElement( MORPH_RECT, Size(5, 5));
            morphologyEx(dst, dst_morph, MORPH_CLOSE, element); // set dst_morph as the output
of morphology
            imshow("dst_morph", dst_morph);
            //=========================================================================/


            ///  Find All Contour   ///
            findContours(dst_morph, contours, hierarchy, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_SIMPLE);
            if (contours.size() > 17) // 최소 contour size를 17으로 하여 멀리있는 대상과 이마나
목이 아닌부위 제외
            {
                /// Find the Contour with the largest area ///
                int idx = 0, largestComp = 0;
                double maxArea = 0;
                for (; idx >= 0; idx = hierarchy[idx][0])
                {
                    const vector<Point>& c = contours[idx];
                    double area = fabs(contourArea( Mat(c)));
                    if (area > maxArea)
                    {
                        maxArea = area;
                        largestComp = idx;
                    }
                }
                // contour부분 bounding box 좌표
                Rect boxPoint = boundingRect(contours[largestComp]);

                ///  Draw the max Contour on Black-background  Image ///
                Mat dst_out = Mat::zeros(dst_morph.size(), CV_8UC3);
                drawContours(image_disp, contours, largestComp, Scalar(255, 255, 255), 2, 8,
hierarchy);
                drawContours(dst_out, contours, largestComp, Scalar(255, 255, 255), -1, 8,
hierarchy);
                namedWindow("Contour", 0);
                //imshow("Contour", dst_out);

                //only contour
                // contour 이미지를 0~1로 normalization하고 원본 이미지와 convolution해주어
```

```
contour에 해당하는 부분만 값을 가지게함
                mul = Mat::zeros(image.size(), CV_8UC3);
                mul = image.mul(dst_out / 255);
                //imshow("mul", mul);

                //SPLIT
                Mat mul_final, mul_split[3];
                split(mul, mul_split);
                Mat mul_v = mul_split[2]; //hsv중 contour에 해당하는 v값
                Mat mul_reshape = mul_v(boxPoint).clone().reshape(0, 1); // 2D -> 1D
                cv::sort(mul_reshape, mul_final, SORT_DESCENDING); //각 countour당 최대 v값
정렬


                int cut = (mul_final.cols) * 0.005; //전체 0.5%  데이터의 수 (소수점 버린값-
>주소값 때문에 정수)
                int T_max = trunc(25 + (double)mul_final.at<uchar>(0, 0) * 15 / (255)); //
최고온도
                double T_sum = 0;// 상위 0.5%를 제외한 후 상위 5%의 합
                int T_avg = 0;// 평균온도
                double T_v = 0; //countour 내 온도값
                int high = (mul_final.cols) * 0.05; //상위 5% 개수
                for (int i = 0; i < (high - cut); i++) //상위 0.5%를 제외한 후 상위 5%의 합
                {
                    T_v = (double)mul_final.at<uchar>(cut + i);
                    T_sum += 25 + (T_v * 15 / 255);
                }
                T_avg = trunc(T_sum / (high - cut)); //평균 온도
                //cout << "T_avg:" << T_avg << endl; //각 countour당 평균 온도

                //최고온도 출력
                string max = "Max:";
                string str1 = to_string(T_max);
                cv::putText(image_disp, max + str1, Point(image_disp.cols * 0.05, 50),
                    CV_FONT_HERSHEY_SIMPLEX, 0.8, Scalar::all(255), 2);

                //평균온도 출력
                string avg = "Avg:";
                string str2 = to_string(T_avg);
                cv::putText(image_disp, avg + str2, Point(image_disp.cols * 0.3, 50),
                    CV_FONT_HERSHEY_SIMPLEX, 0.8, Scalar::all(255), 2);

                if (T_avg >= 38) //38 이상의 온도일 때 Warning 출력
                {
                    cv::putText(image_disp, "WARNING", Point(image_disp.cols * 0.05, 150),
                        CV_FONT_HERSHEY_SIMPLEX, 2, Scalar(0, 0, 255), 2);
                }

                /// Draw the Contour Box on Original Image ///
                rectangle(image_disp, boxPoint, Scalar(255, 0, 255), 3);// contour에서 박스
그리기
                namedWindow("Contour_Box", 0);
                imshow("Contour_Box", image_disp);
                videoWriter << image_disp;
```

```
                ///// Continue Drawing the Contour Box  ///
                rectangle(dst_track, boxPoint, Scalar(255, 0, 255), 3);// contour box가
누적되어 검은 화면에 그려짐
                //namedWindow("Contour_Track", 0);
                //imshow("Contour_Track", dst_track);
            }
            else
            {
                ///  Draw the max Contour on Black-background  Image ///
                Mat dst_out = Mat::zeros(dst_morph.size(), CV_8UC3);
                //namedWindow("Contour", 0);
                //imshow("Contour", dst_out);

                //only contour
                mul = Mat::zeros(image.size(), CV_8UC3);
                mul = image_disp.mul(dst_out / 255); // 원본 이미지에 contour 이미지를
normalization하여서 행렬곱을 해주어 contour에 해당하는 이미지만 출력
                //imshow("mul", mul);

                /// Draw the Contour Box on Original Image ///
                namedWindow("Contour_Box", 0);
                imshow("Contour_Box", image_disp);
                videoWriter << image_disp;
                ///// Continue Drawing the Contour Box  ///
                //namedWindow("Contour_Track", 0);
                //imshow("Contour_Track", dst_track);
            }

            char c = (char)waitKey(10);
            if (c == 27)
                break;
        } // end of for(;;)

    }
    return 0;
}
```