**Classification Static Hand Gestures**

CS 585 HW2

Can Erozer

No teammates

02/15/2024

Problem Definition

In this assignment, my aim was to classify four static hand gestures through my webcam. My algorithm has to find the approximate location of the hand, process the hand image, and analyze it to make a classification.

If the model is improved with increased number of labels and with making it adaptive to various environments, then this model can be used, for example, to transform the sign language into text.

I made several assumptions when starting this assignment. One of them is that I thought I could create my own tools that iterates the images with loops. But I realized that iterating through an image is really time-inefficient, and with the tools created by this way is not fast enough to make classification within seconds. So, I had difficulty when trying to find efficient ways to make image and video processing. The fastness of the functions is essential since my algorithm has to respond to the hand gestures within seconds.

Method and Implementation

My initial motivation sparked with the idea of blob transformation technique that we learned during lecture. I thought I can convert the image into binary image and reduce the complexity of the problem. This is because I don’t perform this experiment in an environment that has a perfectly plain background; and the appearance of surrounding objects could confuse my algorithm, I assumed.

In order to create binary images, I had to do the skin detection. So that the areas that have my skin would be white, and every other thing would be black. I achieved to perform skin detection with working HSV instead of RGB. I opened my webcam and experimented with different HSV values to find the proper range that helps to detect my skin. Here are the functions that I used to do skin detection (mask is the binary image):

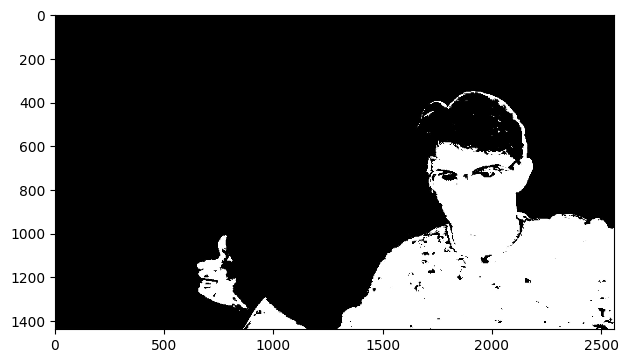
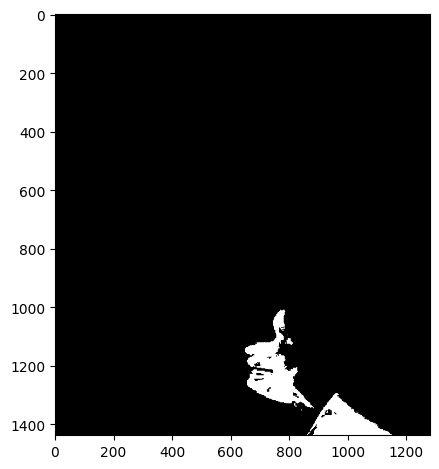
hsv= cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

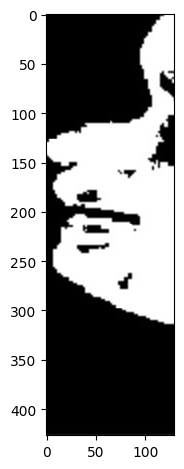
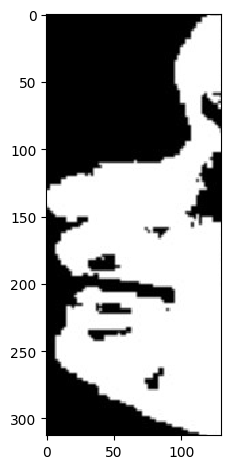
lower\_skin=np.array([0, 60, 0])

upper\_skin=np.array([179, 255, 255])

mask=cv2.inRange(hsv,lower\_skin, upper\_skin)

After, I found how to convert my images to binary images, I prepared the target images which are the four constant images. In order to do that I took screenshots of the four gestures from a distance that is halfway between my camera and the background which is the wall through my camera. And then I processed the photos by first transforming them to binary images, and then I reshaped the images in a way that the whole photo is the gesture itself. Here is the example of these steps: (All the target images can be found at the bottom of label\_images\_preparation.ipnyb file.)

(See label\_images\_preparation.ipnyb for further detail)

To be concise, I first cut the image to half, since the hands are always located at the left side of the image. And then I cut the image from the points that I encountered a white pixel when I am iterating the images from different starting points.

In my second step, where I classified the hand gestures, I transformed the images of video-frames to binary images, zoomed-in a place where the hand gesture is located, and then performed template matching using cv2’s matchTemplate() function with the square difference parameter. I compared the squared difference metrics for each class, and found the smallest of these metrics. The one that yields the smallest squared difference metric becomes the predicted hand gesture.

To create graphical response to the hand gestures, I drew boxes around the gesture and printed the name of the predictions.

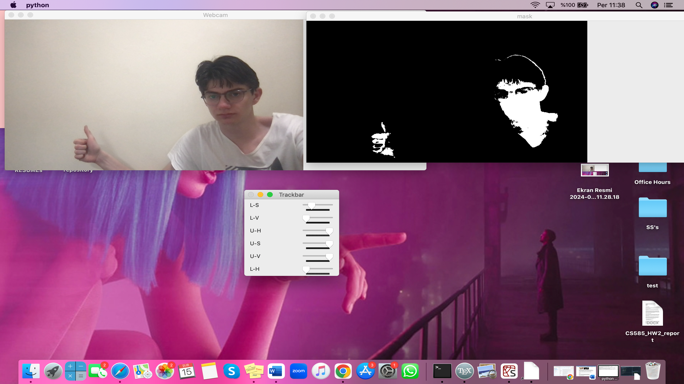
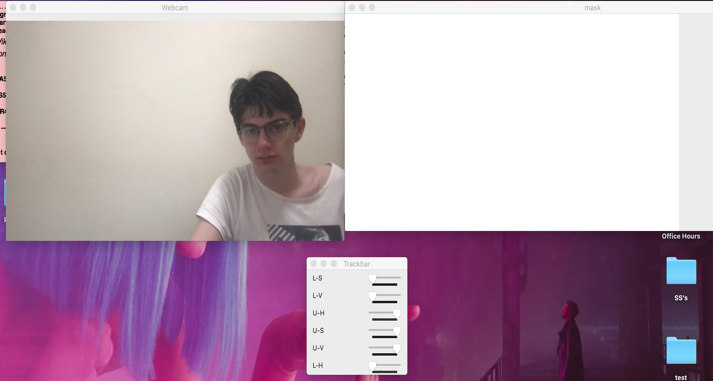
Experiments

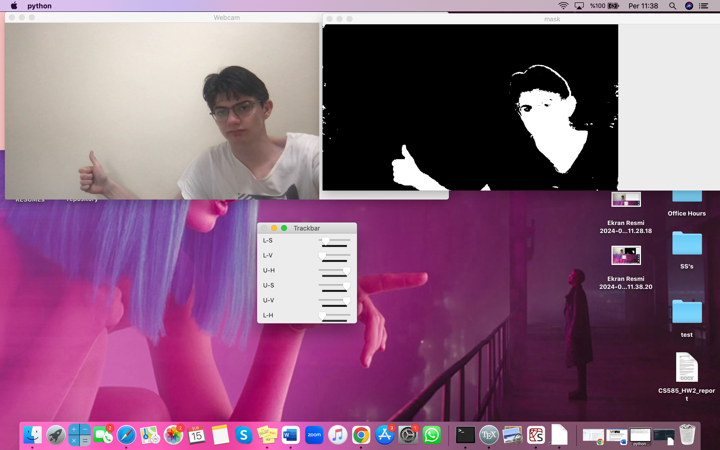
I first tried to implement my own skin detection algorithm. In order to do that, I first took photo of the background which is a whitish wall. And found a range of the RGB values of my wall. But my wall was not perfectly smooth. And the little holes created problems. So, I blurred the image to smooth the image. Using this RGB range, I then tried to transform the images that has the hand gestures to binary image in a way that, the wall is white and every other thing, which is my body, is black. Here is an example of a transformation:



As can be seen above, my algorithm was not able to classify the part of the wall that has shadow on it as the background. Plus, it takes 13 seconds to make this transformation which is very inappropriate for the task of classification through webcam. So, instead of using my own algorithm, I used different ways that use cv2 functions and HSV channels.

Furthermore, I experimented with different HSV values for proper skin detection. To find the best HSV range, I created a trackbar which I can see the results right after I changed the range. Here is an example of me experimenting it:





(Notice the different values of L-S (lower saturation value) in each photo)

In order to test my model, I made the same ten hand gestures consecutively for each hand gesture. So, I had total of 40 predicted classification. I printed each prediction in my terminal from which I took note of the predictions. Here are the predictions:

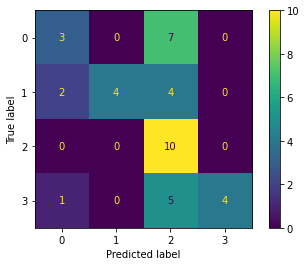
for label\_0= [2,2,2,0,2,0,0,2,2,2]

for label\_1= [0,2,1,1,2,1,0,2,2,1]

for label\_2= [2,2,2,2,2,2,2,2,2,2]

for label\_3= [2,3,2,0,2,3,2,2,3,3]

After concatenating these predicted classes. Here is the confusion matrix:



Using the functions below, I then found accuracy, precision and recall metrics:

Accuracy = metrics.accuracy\_score(actual, predicted)

Precision = metrics.precision\_score(actual, predicted, average='weighted')

Sensitivity\_recall = metrics.recall\_score(actual, predicted, average='weighted')

Accuracy: 0.525

Precision: 0.721

Recall: 0.525

From the confusion matrix, I can make the inference that my model has tendency to predict class 2. The image below is the gesture that belongs to class 2.



The images below are the gestures that belongs to class 0, 1,3 respectively.



I think gesture that belongs to class 2 is the one that resembles the most to every other gesture. Therefore, my model has tendency to predict class 2 most of the time.

Results:

Here are the some screenshots during my model’s prediction of hand gestures:









Discussion:

I think my model succeeded to detect my skin and create blobs from my body. My model is able to make video and image processing and analyzing within 4 seconds. It is able to predict the gestures from class 2 with 100% accuracy.

But there are many parts that my model has to be improved. My model is only able to work properly in a background that is mostly plain. For example, I tried to run the model in the library, and the various objects in the background reduces the quality in predicting the gestures. Also, for better predictions, the hand gestures must be made in a certain distance from the webcam. If the gesture is made in a far place or close place from the webcam, the performance of my model is deteriorated. So, my results show that my method is vulnerable to some limitations.

For potential future work, I could implement image pyramid to make my model able to classify gestures at any distance. I could improve the method that I used for background differencing that my model can focus on me even though the background is not a plain wall. Also, the Hausdorff distance can be used besides template matching to improve the classification performance.

Conclusions:

Based on my discussion, my conclusion is that in order to classify hand gestures that are similar to each other, one should take into consideration of different methods like background differencing, skin detection, template matching and image pyramiding. In order for this model be used in real life some adjustments regarding the adaptivity of the model to different environment must be made.

Credits and Bibliography:

<https://www.youtube.com/watch?v=cDP_4VbC_sE&t=1s>

<https://www.youtube.com/watch?v=FygLqV15TxQ>

<https://www.evidentlyai.com/classification-metrics/multi-class-metrics#:~:text=Calculate%20the%20number%20of%20true,averaged%20precision%20and%20recall%20scores>.

<https://opencv.org/>

<https://nalinc.github.io/blog/2018/skin-detection-python-opencv/>