DSP - Project

Presented by:

Noam Navon - 322937384
Reem Hanin - 207919267
Shay Zvibel - 318872892
Gilad Abukrat - 209203603
Amit Dotan - 207055971
Lior Atal - 316561901
Hadar Sabag - 207919440
Gal Ahrak - 318843125
Ron Hadad - 318553971
Niv Amgar - 208498956
Kfir Borenstein - 322955931
Michael Malkhin - 320882327

Topic: Image Segmentation from the Videos
Lecturer: Dr. Thomas Trigano

Theoretical Introduction

For EMG activity recognition related to the motion of the hand, our group focused on separating the hand from the background, known as segmentation.
The general idea is to make a binary image where the pixels related to the segmented object are seen as usual while the rest are blackened out. This process helps make the picture more useful for different applications, such as detecting object motion, location, size, etc.
There are numerous algorithms for the process of segmentation. We focused on segmentation via color specification.
The original image, stored in RGB format, is a 3-dimensional tensor, where the first dimension is the pixel height coordinate, the second is the pixel width coordinate, and the third is the contribution of each color domain, as exemplified in Figure 1 for a simple 6x6 pixels image.

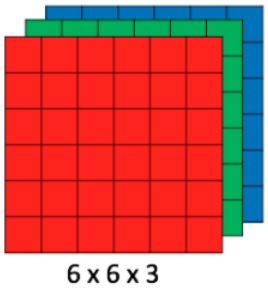

Figure 1

The third dimension of colors can be transformed into other forms that are more intuitive to work with, such as HSV, which focuses on hue (tone), saturation (boldness), and value (brightness), and L*a*b*, where a* and b* are color layers. L is Lightness, which determines how bright the color is. Different color dimensions are often reasonable to represent on a 3D coordinate system, as shown in Figure 2.

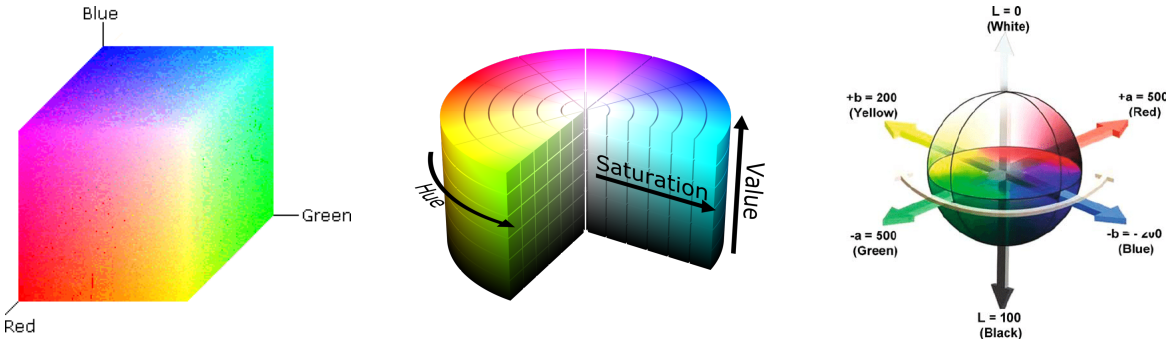

Figure 2

We found that the “a*” layer, hue, and Saturation are particularly good indicators of skin tones when detecting skin colors.

Algorithm

In this section, we will briefly explain each function and how, when used wisely, they deliver the desired results.

SkinColorEnhancement

This function takes an RGB standard frame and, using the Hue, Saturation, Value, and the 'a*' layer (of L*a*b*), deliberately multiplies it to maximize the value of pixels related to the hand.
It is essential to understand that the skin tones in hue are very close to zero; hence, we shifted the spectrum by 180 degrees to obtain a continuous spectrum with which to work.
Finally, the square root of each pixel was taken as a basic intensity transformation since multiplying images whose pixels vary from zero to one results in a very dark result. Histogram equalization is also fine for this purpose.

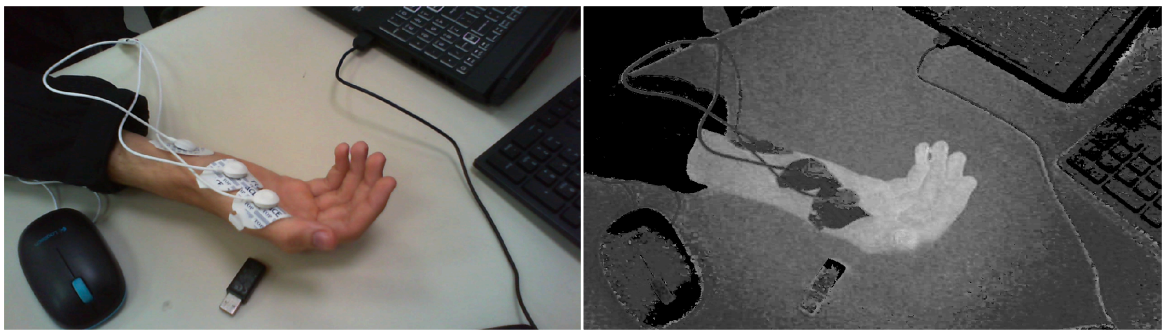

Figure 3

Figure 3 shows an example of skin tone enhancement. Giving the skin-tone pixels high values sets the ground for thresholding.

handHSVstats

Given the ability to enhance skin tones, this function finds the statistics of the skin tone. Starting with a threshold of one, all the pixels are initially filtered, and then we decrease the gate gradually. We want a small number of pixels to remain in the picture. Figure 4 displays screenshots taken by progressively reducing the threshold, stopping when 50,000 pixels pass. When we reach the desired number of pixels, the mean and standard deviation of the pixels are calculated for the hue, saturation, and value, giving back a pair of 3-element vectors.
Assuming that the skin color statistics don't change during the video, it is enough to calculate the first frame statistics only and use it for the rest of the video, thus sparing runtime.

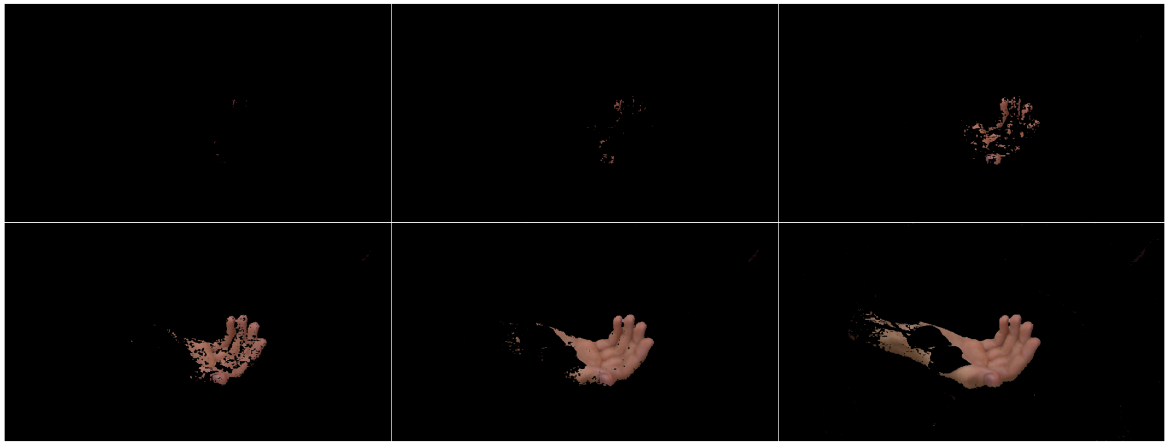

Figure 4

SegmentHand

The skin tone statistics now become the reference for the statistics we want to see in a segmented image. This function goes the other way around and starts with a very low threshold that lets any pixel pass. We gradually increase the gate until the segmented image's statistics are close enough to those of the skin tone, promising a clean cut.

Figure 5 demonstrates the process for the same frame. The background shrinks towards the hand instead of the expansion we saw in the statistical characterization. Comparing the last result images of the two processes, we can see that using this technique, we obtained a bigger, fuller segment, especially towards the arm just above the wrist. The exact process applies to every frame, as the contour of the hand changes when the hand moves.

Figure 5

It is important to clarify what “close enough” means regarding error tolerance when compared to the statistical reference. The users of the product, or the purpose of it, determine the type of the data. For instance, skin samples of people from varied ethnic groups will result in variations in skin tone, just as will cause varying lighting conditions, camera profile, and more. As the data gets more varied, the constraints loosen up, and the algorithm is more prone to noise or wrong thresholding.
Our dataset was imperfect; some hands were captured and zoomed in more than others, and the angle needed to be fixed. In some of the videos, two skin tones of two hands and more inconsistencies can be found. Also, we don’t have videos of people with dark, eastern, or very pale skin, so even though our framework works pretty well, it might be limited to specific skin tones.

Figure 6 is an example of problematic data. The tester’s clothing has bold red and orange strips, which fall in the hue range of human skin. When statistics are measured with these elements, the mean of the hue and saturation is shifted, and their variance increases. These false statistics make the comparison and error tolerance meaningless for the values we chose for good samples; hence, the segmentation doesn’t perform appropriately.

Figure 6

NoiseReduction

For appropriate segmentation, we would like the background to be completely black so that we know that the pixels of the objects identify as non-zero elements; thus, we need to eliminate the noise. Luckily, the noise has some distinct characteristics if the previous stages succeeded; among these properties are:

- The noise is pierced and not uniform in most cases
- The noise consists of zero and non-zero pixels pseudo-randomly distributed in the area
- The non-zero pixels in the noise have a small area compared to the object of interest

Thanks to the first two properties, a morphological element, such as a disc, can go at the edges of the non-zero pixels and eliminate noise in their way. This method will remove much more noise than the object of interest since the noise is sparse; thus, the morphological element captures more of it. The third property just described justifies lump deletion, which is implemented next.

LumpsDeletion

The function gets an image, a boolean mode that determines whether to remove black or white lumps from the binary image and the maximum size of the lamps needing removal. The algorithm is the same for black and white lamps. The ‘mode’ variable goes with an xor operation with the binary image to invert it before and after the lump removal.
The function's theory is to identify a neighborhood of connected pixels from all directions, count the number of pixels in each neighborhood, and, if it is small enough, invert it as required by the mode parameter.

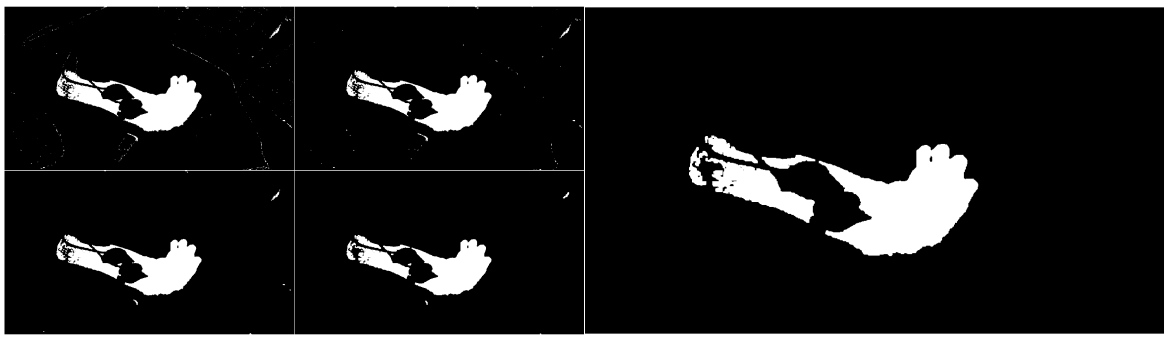

Figure 7

Figure 7 is an example of successful noise removal. The image is binary for accessibility since a small group of pixels is not perceivable when the contrast with the background is minor. The four small images on the left show the process of cleaning sparse noise with a disc and the big lumps that remain at the end are removed using the lump deletion algorithm.

main

The main function adds the data to the path and executes everything step by step. When the final binary segmented image is at hand, simple multiplication with the RGB frame gives the segmented frame.
If a rendered real-speed video is required, the VideoWriter class can be utilized to add frames to a video file, as we have added in the commented-out code. We have added a rendered video for reference.

The execution time of the code obviously depends on the hardware, so real-time performance can't be guaranteed. While running on a 2019 Mac with an Intel quad-core i5 and 8GB RAM, this code renders a frame in 0.65 seconds. Assuming 30-FPS, it takes approximately twenty seconds to render a second of video, so real-time rendering is not possible with such hardware.

One last thing to consider is using the "a*" color domain statistics to set the threshold, too. Since it significantly affected skin tone enhancement, it might have a similar effect as a stable statistical criterion.

Conclusion

Segmentation can be done with different methods, like spatial filtering to enhance contours, analysis in the Fourier domain, focusing on the skin's texture, and more. We also saw that color specification is a straightforward approach that works pretty well for stable, predictable conditions.
The theory implemented in this algorithm is not directly related to the things taught in class but rather an expansion of those into the theory of digital image processing.

Further development of the segmentation should involve separating different parts of the hand, resulting in distinct segments for each finger separated from the palm and wrist, and, even better, segmentation by joints. Those are much easier to implement when the hand is already segmented from the much more chaotic background. The next level of segmentation would give more practical files to work with when combined with the EMG recordings, as specific object tracking allows more flexibility with predicting motion from EMG pulses and vice versa, which will pile up to high-quality data ready for neural network training.

We hope our contribution of the segmentation algorithm to the EMG signals and finger movement relation project will result in success in the research.