Visual Combination Lock

Federick Gonzalez, fag2113

February 2020

1 Design Specifications

In this program, a short sequence of three images is taken and analyzed in order to find a predetermined sequence of locations and poses. The photographs were taken with a Canon DSLR camera to ensure decent quality and to take raw CR2 files in case they were needed at some point. Images were taken on various backgrounds, until it was decided that a blue background would provide the best results for this kind of program. The Python version of OpenCV was used to complete this project using PyCharm, a surprisingly helpful IDE.

2 Domain Engineering Step

As stated above, a Canon DSLR camera was used to obtain images, which were stored as both JPG and CR2 files in the images folder. After the primary set of images were taken against a black chalkboard with bright lighting in the background, it was determined that the reflectiveness of the chalkboard resulted in color that matched the washed out skin too closely. This resulted in an inability to accurately detect whether or not skin is present, so different backgrounds were then tested. It was opted to use a less reflective blue surface to better capture the difference between the skin and the surrounding area. The final chosen background was the blue fabric of a couch in Schapiro Hall. Left hands were used in this project, and the user is required to wear a non-skin colored sleeve to prohibit their arm from being detected. In particular, a black sleeve was chosen, to ensure that nothing other than the hand could be viewed as skin color.

Over the course of a few days, 58 images of hands in different positions and forming different shapes were collected, and out of these 58, all were used in testing, but the system was optimized for the last 37, as they did not record the inconvenient blackboard reflection, which skewed results. A final, completely unrelated picture was also included.

The images are stored in several directories. There is a generic directory, *./images*, which stores all the pictures that are to be used. Then there are 12 separate directories for each test case, labeled 1-12, and within each of these directories, there are images labeled *hand1, hand2,* and *hand3*, in order to allow for simple processing of images through loops. To test more images, simply add a new directory, name images according to the convention above, and

3 Data Reduction Step

In order to determine the region of the color space that is skin, BGR (The default color space of OpenCV) was converted to YCrCb. A lower bound, (0, 133, 77) and an upper bound, (235, 173, 127) was determined to give the best estimate for skin color. the YCrCb color space proved to be more effective than the HSV color space, discovered by testing a series of 58 images. Using these upper and lower bounds, an image was created by taking only the pixels that fell between these two bounds, using the OpenCV inRange function. Now that an image existed with only the pixels that were roughly skin colored, the edges were eroded and dilated, and then finally a Gaussian Blur was added to smooth the edges further. The mask was then placed over the original image, and the bitwise and operator was used to include pixels only present in both images. Once it was determined that images and hand positions could feasibly be captured, it was time to start determining grammar.

To determine positioning, the image was divided into nine sectors, named $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. Each sector takes up $\frac{1}{9}$ of the images total area, and it can easily be visually determined in which sector a hand would be in using the draw_lines() function, which using relative image size would draw one line across $\frac{1}{3}$ or $\frac{2}{3}$ of both X and Y. Then, it was made to detect where the moments of specific contours lay within the image, printing out a quadrant number to the console.

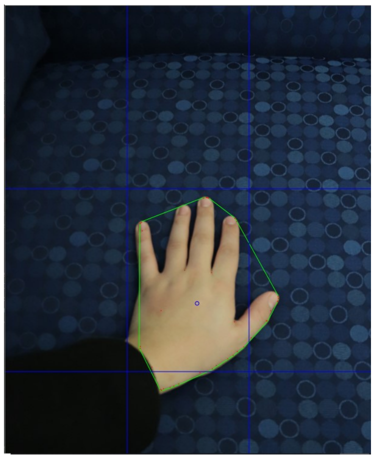


Figure 1: The 4 splayed position

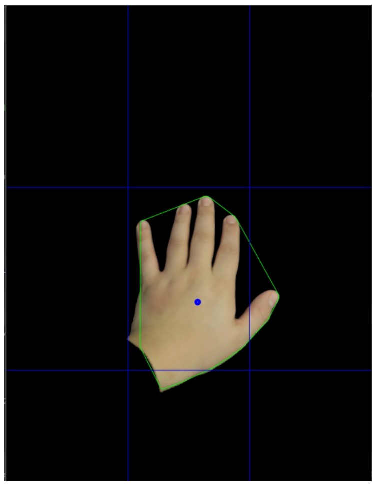


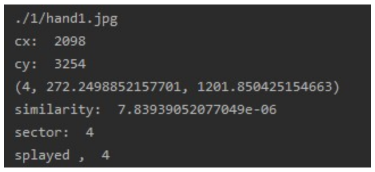
Figure 2: Repeated with skin mask

Finally, when the assignment was described, a splayed hand and a fist were mentioned, and they were used for the beginning of this program. A ideal model for a splayed palm and fist were created and their contours were discovered. Then the contours of current images were checked against these predetermined images, using OpenCV's matchShapes method. The matchShapes method finds the similarity between objects using the Hu Moments of two different contours, which are essentially center moments that are invariant to any transformation other than reflection. There are three methods to calculate similarity using Hu Moments, and the following was chosen:

$$D(A,B) = \sum_{i=0}^{6} \frac{|H_i^A - H_i^B|}{|H_i^A|}$$

This method seemed to give the most accurate similarity value, as for the ones that seemed visually similar had a similarity value of $similarity \leq 1$ and those that were dissimilar had a similarity value of $similarity > 1$.

If their similarities were less than one, the two hands were deemed similar enough to clarify whether a hand was in either the **splayed** or **fist** position. When the display feature is activated, a window pops up similar to what can be seen in figure 1. The gridlines are clearly visible, as is the moment and the contour surrounding the hand. The underlying colored image after the mask is applied can be viewed in figure 2. The sample output printed to the console can be seen in figure 3.



```
./1/hand1.jpg
cx:  2098
cy:  3254
(4, 272.2498852157701, 1201.850425154663)
similarity:  7.83939052077049e-06
sector:  4
splayed ,  4
```

Figure 3: An example of the 4 splayed position output.

To explain this output, the first two values of *cx* and *cy* represent the center, or moment coordinates of the current hand's contour. Then follows location info, which shares the sector number, the distance to the center of the sector, and the maximum possible distance within a sector from the center. Finally, it displays sector again, so it's easy to see what sector it is, and finally there is a tuple which shares the detected feature and the detected location.

4 Parsing and Performance

To handle the grammar, there are a few simple cases. The screen was divided into nine segments, as described above, and each segment is labeled $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. The sectors are arranges as follows:

0	1	2
3	4	5
6	7	8

Table 1: The position of sectors in each image

There are also 3 predetermined hand positions, labeled *splayed, chop* and *fist*. The *splayed* configuration can be viewed in figure 2, and the *chop* and *fist* configuration can be viewed in figures 4 and 5 respectively.

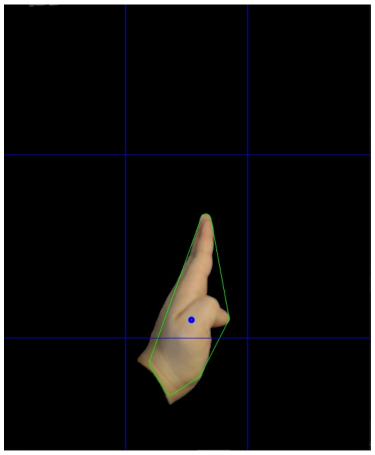


Figure 4: The chop position

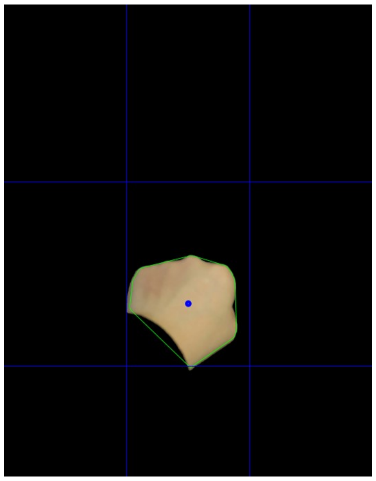


Figure 5: The fist position

As can be seen in each of these examples, there is this small blue dot visible. This blue dot indicates the moment of each contour, which allows the program to determine which sector of the image a hand is in. All of the displayed images above are in sector 4, which is the center of the image. The hand does not need to necessarily fit perfectly within a sector, it only needs to be roughly within the quadrant, and therefore it allows the moment to fall within a specific quadrant.

The sequence is determined successful if the boolean value *valid* is returned as $True$. If there is any deviation from the predetermined sequence either by position or by hand shape, then the boolean value is set to and therefore returned as $False$.

In order to test the program, the following sequences were used:

1. Proper test cases
2. Proper test cases with different images
3. Proper test cases with different images again
4. Proper test cases with Trick Image (backwards palm instead of splayed)

5. Proper test cases, wrong order
6. Proper test Cases with Trick Image (sideways fist instead of normal fist)
7. Proper hand positions, wrong shape
8. Proper hand shape, wrong positions
9. Wrong hand shape, wrong positions
10. Right positions, wrong order
11. Image without hands
12. Repeated image

In order to test wrong hand shape, there was a combination of the correct hand shapes in different positions, or a new hand shape which was deemed the "spider man": This was chosen because it was the hand position that was

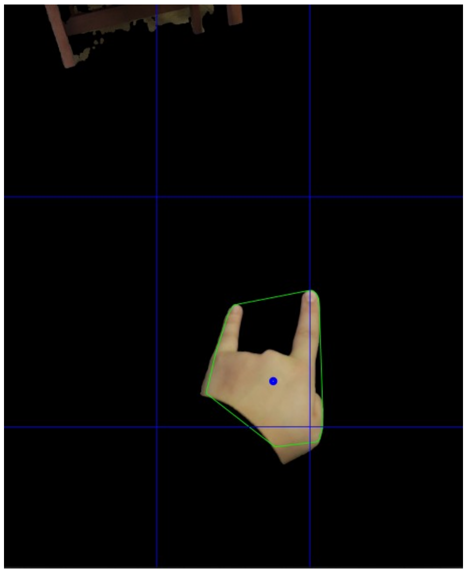


Figure 6: The spider man position.

deemed most similar to all 3 other recognized positions, seeing as the shape can be viewed as a splayed palm, a fist, or the chop if one finger is detected and not both.

Out of the 12 test cases, the first 3 were expected to be true, and the rest were expected to be false. However, the $2^{nd}, 3^{rd}, 4^{th}$ and 6^{th} cases did not work as expected. The second and third were false negatives. There are a few possibilities for reasons these did not work. For the second test case, the fist was held at a slightly different angle than that of the model hand, therefore it was determined to not match the chosen shape, even though visually one might say that it does. Looking at the displayed output, the similarity for the palm shape in the third test case surpassed the similarity threshold of one, and this is likely because the fingers on the hand were spread too far apart for them to be accurately detected by the contour finding method.

The fourth and sixth cases were false positives. In the fourth test case, the splayed palm was flipped over and facing towards the camera rather than away. The shape of this is too similar to the original splayed palm, therefore it was not marked as different, and it passed through all tests. In the final failed test, the fist was tilted similar to the second test case, however in this example the degree of tilting was more akin to rotation than actual tilting, and it was determined to be similar due to computation using Hu moments rather than actual orientation and size.

The non-hand image used in test 11 was as follows: This image was chosen

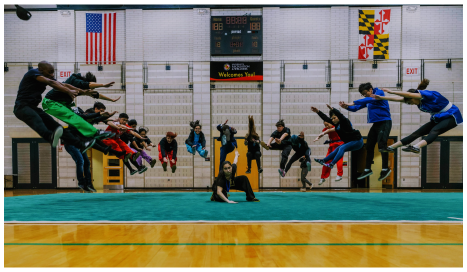


Figure 7: This image has multiple skin-colored pieces.

because it has many different skin colors and any part could be chosen as the hand, to see if the program can protect against non-hand shaped object being matched to hands. The test passed, by rejecting the image sequence.

5 Creativity

Rather than using two images, one with a “fist” “in the center” and one with a “five!” “in a corner”, this project was specified even further. Instead of center or a generic corner, the screen was divided into six sector, and rather than being in a corner, the visible hand must be in a certain sector. In addition, rather than only using a ”fist” and a ”five!”, the hand positions of ”splayed”, ”chop”, and ”fist” were used, in this order. The predetermined sectors where these should be were determined to be $\{4, 6, 8\}$, which corresponds to center, bottom left and bottom right. The spider man position was also detectable, but intended to see if the program could be confused. It was determined that the tests never failed when the spider man hand appeared, therefore it was able to detect this position was different from all other positions.

In addition, rather than solely looking at the sector, the distance to the center of each quadrant was analyzed. If the distance to the center of the sector was greater than $\frac{2}{3}$ the total distance possible from the center, it was deemed too far and therefore likely not in the proper position. As can be seen in figure 3, the distance to the center is 272.45 and the maximum distance to the center is 1201.85, therefore this hand passes the test and is able to be viewed as in the correct position, as the user was likely guessing rather than knowing where the intended position is supposed to be.

6 Conclusion

OpenCV was an incredibly useful tool in the processing of images, and using it I was able to find moments, contours, and a skin mask image. In general, the program seemed to work as intended. 66.67% of the test cases returned the expected result, and a hand gesture specifically intended to fool the program did not work. Using Hu moments and contours based off of an image containing only pixels within a certain color range, the program was able to compare a sequence of images to a predetermined sequence of gestures an positions. The three gestures used were a splayed palm (ref 2), a chop (ref 4) and a fist (ref 5). Each hand was compared against these three reference images, and then their position was found, detailed in table 1.