Right Hand Shape Recognition

Jeremiah J. Flowers and Dennis J. Austill

Abstract—This is a system for hand recognition built for the right hand only. We have developed this system to imitate what a real world hand recognition system should be. This project could be thought of as a template for a more involved system to build upon. Common constraints in other hand recognition system are the use of pegs to hold the hand in perfect position for the image to be process. We relax the constraint and remove the pegs but still ask that the hand be fairly straight and fingers spread out.

Index Terms—No pegs, feature extraction, binary segmentation, finger width

1 Introduction

THIS system follows three approaches used in other systems for the following; image segmentation, hand extraction, and hand verification.

Before working on how to segment the image from the camera, the first question was how we would get the image contour. After asking and then answering that general question, we knew we wanted to have the image be segmented into a binary image. Binary for two colors; color one, the background and color two, the hand. For our system we make the background a single color that we have chosen.

The next step after getting the image we want is to extract the hand shape contour. The idea of contour extraction is absolutely not a new idea but the approach I used is my own idea (I did not follow any method). Because of the binary image I would say it is a non-complex approach compared to other approaches, a.k.a. active shape model. There is also feature extraction that happens while we are getting the contour. We call feature extraction landmarks [1].

The final step is, after getting all the data from hand shaped contour, hand verification. Using the data we use hand geometry to measure the hand. Then test it against the database to verify a match or to enroll a new candidate.

2 Method

Hand are very predicable in shape and do not change much from person to person. Knowing this we used feature extraction to find points of interest and use them to calculate the hand geometry. Our hand geometry only measures the five fingers and nothing else. We do nothing with the contour and nothing with the palm of the hand.

2.1 Equipment and Constraints

To program the code we used Matlab. Matlab can easily manipulate the image matrix using only a few commands and any matrix multiplication for translation would also be a simple command. We use a webcam for taking the image. It is directly connected with Matlab so there is not an extra step between the two. For holding the camera in place, we use a ~~clear box~~ tripod. The tripod holds the camera on top with it face straight down. The camera is facing the background sheet. The distance is always the same, so the having to do any scaling is unnecessary. Since this is shape matching not having to scale removes possible problems with matching. For example, if you have two squares that are different sizes. If you use scaling to make them fit each other they will, even though they were two different shapes.

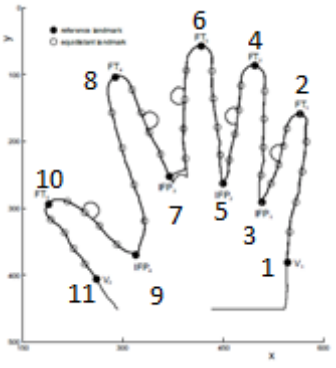
2.2 Image Segmentation

We use background subtraction. Since we know what the background is going to be, green, it is a simple process. Using RGB, the algorithm checks if that pixel has a higher green than red and blue, G<B<R. If the pixel looks green then give it a value 0(black) and if is not green then give a value 1(white) Using a graph of the different skin colors, I noticed that all skin colors leans more to favoring having a higher red than blue/green and a higher green than blue. So any background used should

have as low as possible red value. After going through the all the pixels in the image, it will come out to be a black and white image of a hand.

2.3 Feature Extraction

While using contour extraction, we make its goal while executing to find the features and not actually forming the hand shape. Instead of features we call these landmarks. In the picture, you can see the landmarks are the bolded points at the tops and bottoms of the fingers.



There are eleven landmarks and we skip finding the 1st and 11th for the first pass. Starting at the bottom right of the hand the contour will move up the right side of the hand until it finds landmark 2. Then change directions and move down left until it finds landmark 3. This repeats until the 10th landmark is found. Finding the last landmark we move left down on the thumb until the x-position is the same as the x-position on landmark 9. Back tracking to find the 1st landmark we do the same but want the y-positions for the 1st and 3rd landmark to be the same.

The binary image makes it easy to check what is hand and what background is. There are two processes to moving the contour; RightUp and LeftDown. RightUp looks to the right (3 o'clock) first then turns counterclockwise to the top left (10:30). LeftDown looks to the left (9 o'clock) first then counterclockwise to the bottom right (4:30). Two lists that have coordinates for RighUp and LeftDown tell the contour where to move next. The coordinates are a 5x5 and 10x10 matrix with only the outer borders being a region of interest. This whole process allows us to move around the outer edge of the hand while finding landmarks. Every time we find a landmark we switch from RightUp or LeftDown to LeftDown or RightUp. This process develops a constraint that requires the hand to be right side up and not rotated any which way. As a rule of thumb, as long as the middle finger is close to being straight, then the hand whole is straight enough.

2.4 Hand Geometry

After we have all eleven landmarks we build boxes around each individual finger. The boxes are parallelograms; reason being that fingers are slanted and parallelograms can accommodate unlike square's 90 degree angles. The little finger would use landmark 1, 2, 3 to build its box. What we want are two vertical lines beside the finger, so they can be used to measure the width of the finger. In Matlab, a function called linspace will create a line made up of points (xy-coordinate points). We made this function create those vertical lines besides each finger.

We measure the width of the finger by measuring 150 slices throughout the finger. The 300 points in the two lines create lines between each other, horizontal lines. Next we traverse the horizontal lines and measure how wide that section of finger is. Simple by check the image, pixel value; 0 = background and 1 = hand. We can get the width of the finger in 150 slices from top to bottom.

2.5 Matching

For each hand that gets computed by the system will be defined as 150 points of data. The way we compare and match hands is by taking the difference of the two hands we're comparing. Let's say we compare two images of the same hand, after we take the difference of the two, the new 150 data point values should be close to 0 to 10. In our system we set a threshold of 3. Out of the 150 points its counts the number 3s or less in the data. Next we set another threshold of 110. Out of the 150 points if there are more than 110 points that are 3 or less then we can say that these two hands are the same. So again after we subtract the two hands' 150 point data we test it with two thresholds for; how close the data values have to be and how many of those data values are enough to say these two hands are the same.

3 Results

We do not have results. The only hands that were tested were ours. We kept running into issues with the system not always working with just our two hands.

The ROC curves and the false-positves, true-negatives, etc. can be manipulated with the thresholds in 2.5. Depending on the number of users using the system the thresholds would have to adjust accordingly to find an optimal point.

4 Discussion and conclusions

The next step would be making the system work for both hands and not just right hands. A possible future plan would be to make the system that work on smartphones.

As for how we approached building. We followed [1] and [2] ideas on these papers. We wanted to build a peg free system and the idea of the landmarks would be key idea to implement. The system as whole compared to what I've seen is non-complex mathematically and that is both a pro and con. Its easily affect by noise but requires less computation strength.

References

[1] Alexandra L.N. Wong and Pengcheng Shi, "Peg-Free Hand Geometry Recognition Using Hierarchical Geometry and Shape Matching," Department of Electronic and Electrical Engineering Hong Kong University of Science and Technology

[2] Raymond Velduis, Asker Bazen, Wim Booij, and Anne Hendrikse, "Hand-geometry Recognition Based on Contour Landmarks," Sign and Systems Group, Dept. of Electrical Engineering University of Twente , Enschede, The Netherlands