Feature Extraction of Character Recognition

Pattern Recognition Assignment 2

Rendani Mbuvha, Huijie Wang
rendani@kth.se, huijiew@kth.se

September 2016

1 Introduction

In this assignment, we designed a discrete feature extractor for *on-line character recognition*. The function of the extractor is to draw valid features from the outputs of **DrawCharacter** function.

The output of **DrawCharacter** function is a $3 \times N$ matrix containing mouse information triplets. Each column of the matrix is a 3-dimensional vector of $position\_x, position\_y, pen\_down\_sign$, where the first two elements describes the position of the point on the canvas and the last element indicates whether the user was drawing, and 0 elsewhere. The vectors are sampled from the drawn character according to time-line. An example of the visualized output is shown as Figure 1.

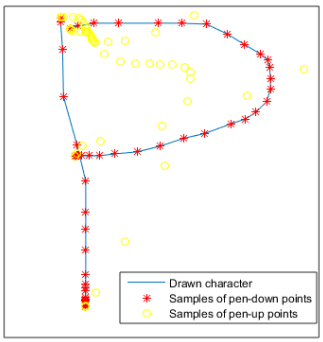

Figure 1: An Example of Observed Character

As can be seen in the plot of the output, the blue line is the trace user draws on the canvas. The red stars are the samples gained from pen-down trace. And the movements of mouse without clicking is also recorded, which are shown by yellow points. It is obvious that these yellow points are not related to the character and show no informative pattern.

Since the output of **DrawCharacter** function is mainly based on the trace of mouse, the pattern is largely influence by location and size, as well as the speed user draws. In the next part we introduce a feature extractor to extract a simple pattern of characters.

2 Feature Extraction

In this section, we introduce the design of feature extractor. Since the features of the character should be identical for similar patterns, using relative location between each neighbours of sampled points is an ideal method expressing the features. The encoded feature classification is shown as Figure 2.

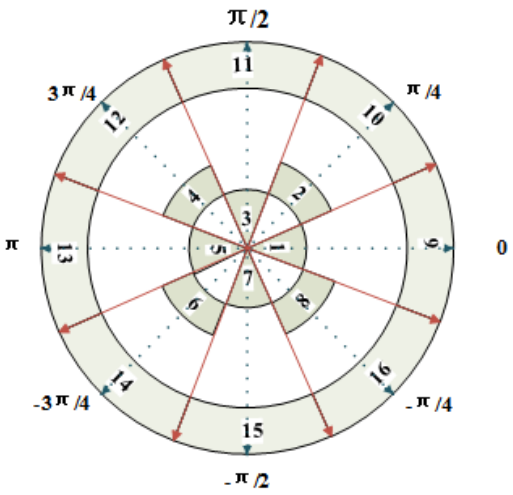

Figure 2: Classification of Directions used as features, where feature 1 to 8 represent pen-down features and feature 9 to 16 represent pen-up features.

As can be seen in Figure 2, 8 directions with 16 features classes are included. Such the features are designed according to 3 factors: location and size character, and pen-down/pen-up movement.

1. Location and Size of the Character

   Since the samples of emissions are represented by location of canvas, a vector can be calculated as the direction between neighbour samples, i.e:

   $$direction = (x_{t+1} - x_t, y_{t+1} - y_t)$$

   From the direction the angle of the vector can be calculated, through Matlab function:

   $$\textbf{ang = angle(complex(direction))}.$$

   As we are only focusing on discrete features, the continuous angles are encoded into eight directions, namely direction 1 to 8. This method removes the influences of various location and size of a character as the features are only based on relative positions between points.

2. Pen-up Movement

   It is obvious that the pen-up movement has no influence on the character since it doesn't leave any trace on canvas. Two kinds of pen-up movements are included: Firstly, the movement before and after drawing. Such observations are simply removed. Secondly, the pen-up movement between strokes. The only relevant factor we need is the relative position between two strokes. However, the pen-up factor should also be recorded: Thus, another 8 features, namely direction 9 to 16 are introduced in order to record the pen-up movement trace.

As a result, the feature example shown in Figure 1, can be encoded into a vector { 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 11, 1, 1, 1, 1, 1, 1, 8, 8, 8, 8, 7, 7, 7, 7, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5 }. Using **imagesc** we can see the pattern intuitively as Figure 3.

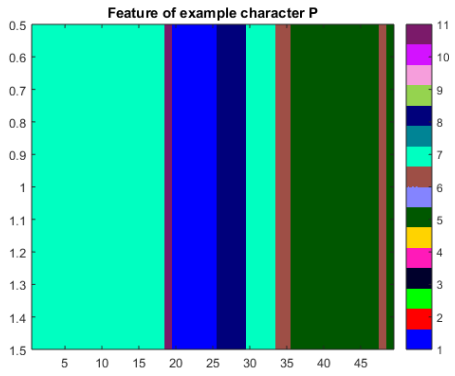

Figure 3: An emission magnitude of feature

3 Experiments

3.1 Test on character "P"

In order to test whether our feature could display invariance to the size or location of the character, we draw the character 'P' in different locations on the canvas and with one which is relatively large in size. We visualize the resulting feature vector using an image of the emission magnitudes.

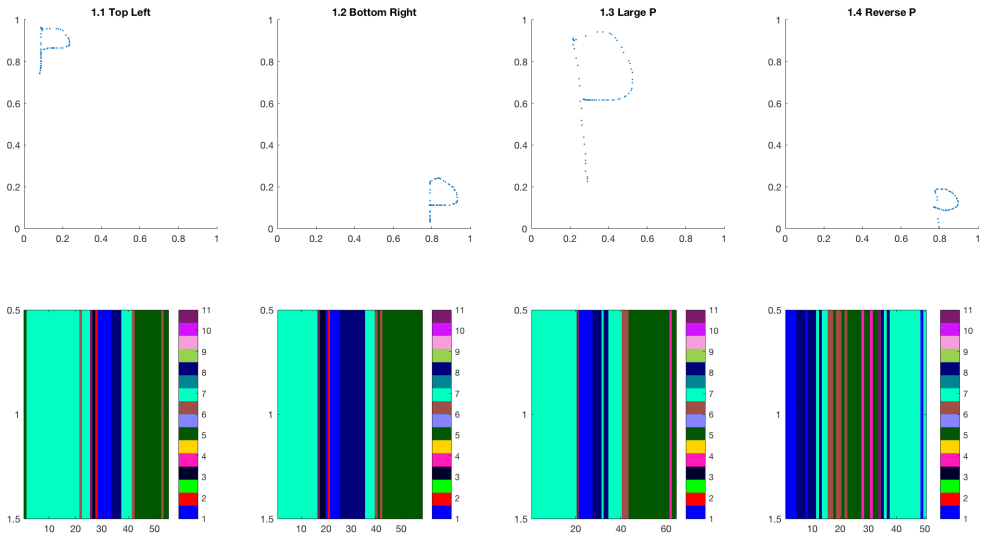

Figure 4: Plot of P in different locations and Sizes, with colour showing feature value and 'x-axis' showing the time sequence

As can be seen in figure 4, the color pattern of the features have a similar stable pattern for the first 3 character drawings, which shows that the chosen feature is invariant to size and location. The emission sequence for 'P' seems to be '7-11-1-8-7-6-5' consistently with some noises.

It is important to notice that the 'P' in last image in 4(1.4) was drawn in reverse - first semi-circle and then vertical stroke - thus, this feature pattern is also a reverse of the pattern shown in the previous images.

3.2 Test on character "T" and "+"

We test whether our feature shows clear differences that can distinguish between close looking symbols like 'T' and '+'.

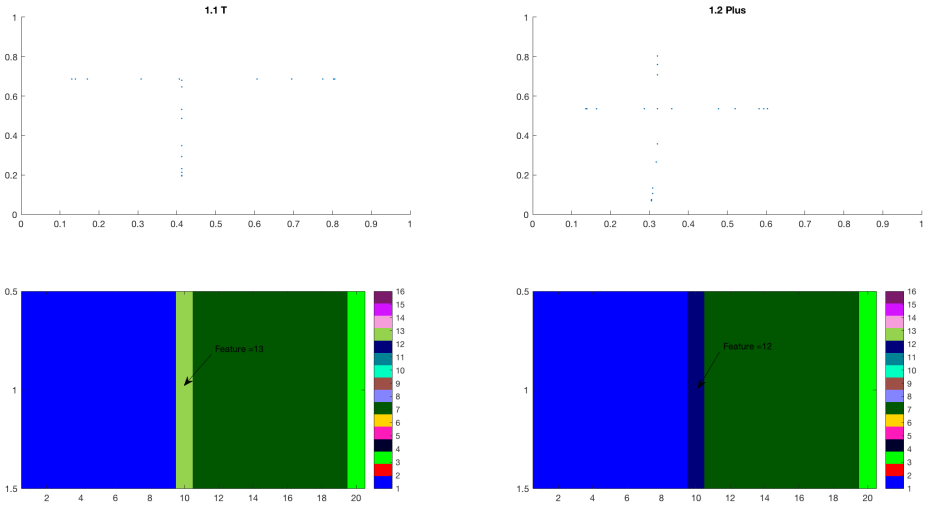

Figure 5: Plot of features from 'T' and '+', with colour showing feature value and 'x-axis' showing the time sequence

As can be seen in Figure 5, the key difference is in emission that happen when the pen jumps after the initial straight line. For 'T' the feature value jumps to 13 since the next line is below and for '+' the feature value jumps to 12 as the next line above the first straight line. This provides a distinguishing feature between '+' and 'T'.

3.3 Do the features differ between characters?

We test to see if our feature extraction algorithm gives different values for different Characters. We observe the features generated from the characters 'A','C','W','X' and 'Y'. From Figure 6 it can be seen that the generated feature patterns are different and can be discriminated between.

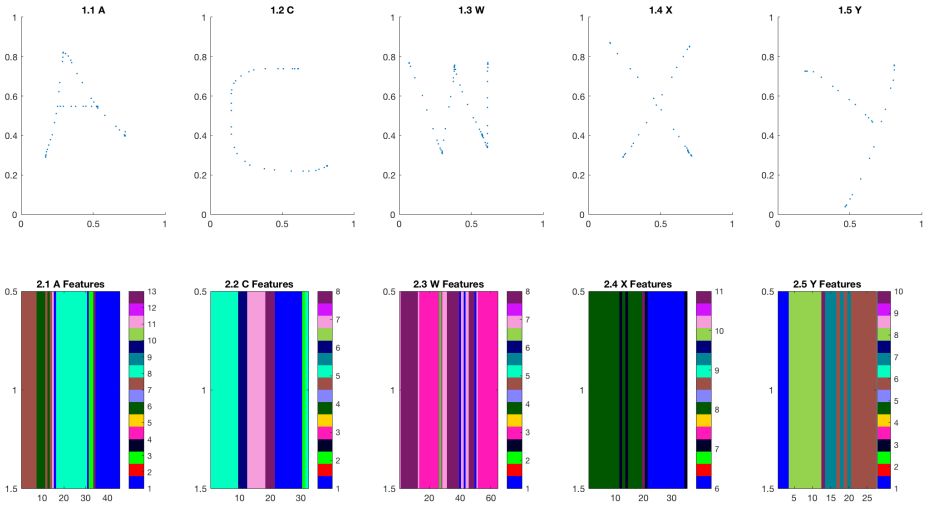

Figure 6: Plot of features generated for various characters 'ACWXY', with colour showing feature value and 'x-axis' showing the time sequence

3.4 Test with Wild Mouse Movements

We tested our feature extraction method for robustness to wild mouse/path movements when the pen/mouse is lifted. In this case when drew the character 'A' with and without wild mouse movements. Ideally we would like the the wild movements not to affect the features, as the recogniser is only interested in the patterns made when the pen is down.

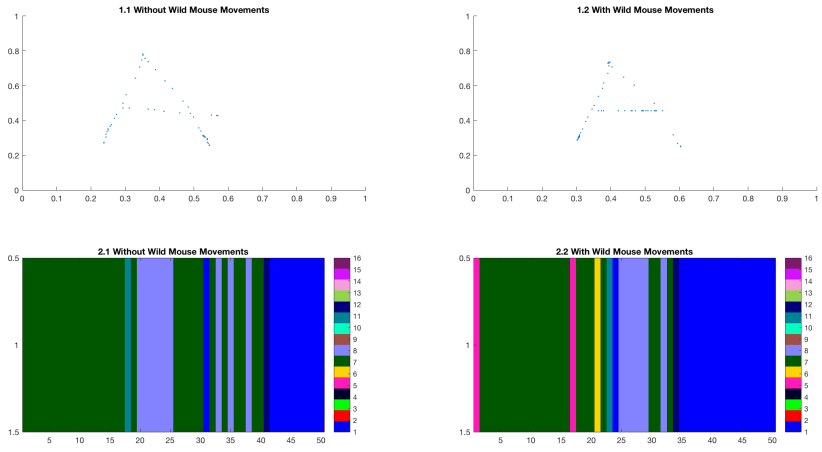

Figure 7: Plot of features from 'A' with and without wild Mouse Movements', with colour showing feature value and 'x-axis' showing the time sequence

As can be seen in figure 7 main feature patterns between the two cases are the similar with some noise as expected. The main feature jumps are between '7-11-8-12-1'. This feature signature does not seem to be affected by the wild mouse movements.

4 Discussion

As we have shown our feature extraction algorithm shows some positive characteristics. One case where the method does not perform well is when the same character is drawn in reverse. This is be cause the features are reliant on the time sequence of points in time. We believe that this would be a problem for most online-recognition methods.From figure 8 it can be seen that the resulting features from 'P' drawn normally and in reverse are significantly different.

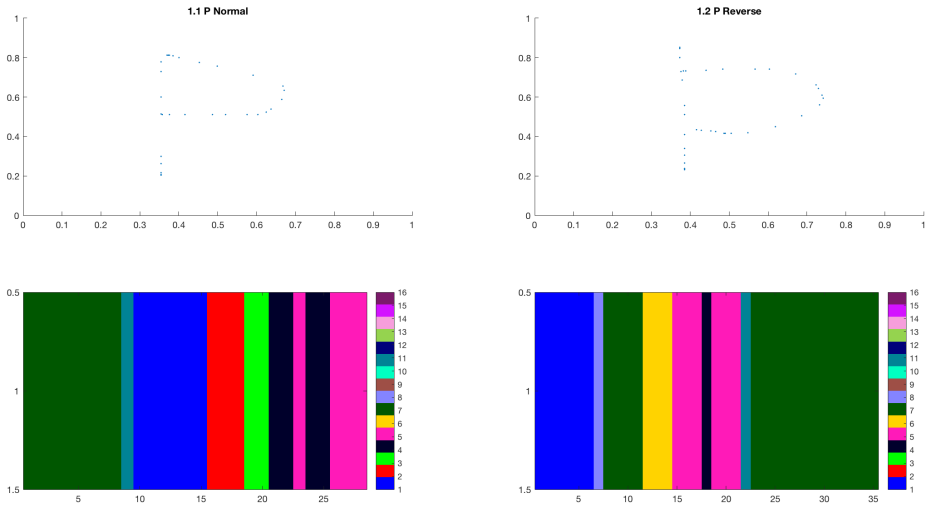

Figure 8: Plot of features from 'P' drawn normally and in reverse, with colour showing feature value and 'x-axis' showing the time sequence

In addition to drawing the same character in reverse another to confuse the system can be to draw the characters with too many unnecessary movement when the pen is down. This can create too much noise that can lead to recognition errors. However additional intelligence is added by using a Hidden Markov Model - which is capable of detecting the true signal amongst random noise.

As this feature extraction method is size invariant - it cannot fully distinguish between uppercase and lowercase of some characters - this could be a problem for characters like 'C' and 'c' or 'O' and 'o'.