Assignment 2
Character Recognition

Thomai Stathopoulou (*thomai@kth.se*)
Dario Vidas (*vidas@kth.se*)

October 11, 2015

Feature extraction design

This section describes the procedure of extracting features for drawn characters. The input is a sequence of 3D vectors. Each vector represents a screenshot of the path and status of the mouse, while drawing a character on a fixed plane. This means that, each vector contains information about the position (xy coordinates) of the mouse (first and second element of the vector) and the mouse's status, 0 if the mouse is not drawing and 1 if the mouse is drawing (third element of the vector). The screen-shots are taken at a fixed frequency.

Feature extraction has to follow certain rules and satisfy certain criteria, such as the speed of the extraction procedure, the dimensionality of the resulting feature, robustness of the feature regarding certain changes of the input sequence, etc.. This means, that we need to extract features that are as compact as possible, that contain as much information as possible and the extraction has to be as fast as possible. All of that has to be taken into consideration, while designing and implementing a new feature. The design of the feature described in this section tries to satisfy as best as possible most of these criteria:

- After receiving the input described above, the first step is to filter out all the vectors representing the path of the mouse, while it is not drawing, i.e. the time when the pen is up before starting, or after ending the drawing of the character and while drawing, when the pen needs to "jump". This immediately reduces the dimensionality of the feature from 3D to 2D, since the third element of the vector is no longer needed (there are only ones left).

- We now have a sequence of 2D vectors, representing the xy coordinates of the pen, while it is drawing. At his point, we remove consecutive vectors with same values. These vectors bare no important information, since they only show that the pen remains stationary, while drawing, which is considered to be unimportant information about the drawing of a character.

- In order for the feature to be independent of the size of the drawn character, we normalize all the coordinates to the height of the character. That way, the resulting coordinates form the corresponding character with height always equal to 1 and width depending on its original form.

- Now we want to make our feature independent of the position of the character within the plane. This happens by translating the character, so that its leftmost and bottom pixels touch the plane's left and bottom border respectively.

- The next step is to calculate the derivatives dx and dy of the coordinates, so that we keep the information, of how much the position of the pen has changed between two consecutive frames. This way, we emphasize the information of large jumps of the pen, which we think is a very important part of the feature.

- We the calculate the angle (using MATLAB's atan2) and the distance between two consecutive frames. Eventually we get the pen's velocity's vector (first element is distance, second element is angle).

- At this point, even though our vectors are very representative, they have two aspects, which we want to change. They have real and not discrete values and they have large fluctuation between consecutive frames. For this reason we chose to quantize both values (angle data into A_n bins and distance data into D_n bins). This corrects both aspects, by turning the real values into discrete and by smoothing the fluctuations.

- We next perform an extra alteration to the distance value. At each step, we assign the value of the distance of the most recent jump. This way, if there is a change of speed during a stroke (resulting in a larger distance between the consecutive frames), it will be disregarded, so that we only keep the information from when the pen leaves the paper and touches it again.

- We finally map this 2D discrete data $[angle, distance]$ into a 1D feature vector using the following linear equation: $f(t) = A_n \cdot (d(t) - 1) + a(t)$. With this, we create a discrete feature vector with values within the range $[1..A_n D_n]$. This discrete feature vector can be represented by the discrete distribution implemented in the first part of the project.

NOTE: For the following figures we used $A_n = 8$ and $D_n = 6$ for the quantization of the angle and distance, respectively. The value $A_n = 8$ was chosen, so that the angles are quantized at 45^o intervals. The value $D_n = 6$ was chosen, so that the distance values can be smoothed out. These values can be fine-tuned at a later stage.

Similar sequences of the same character

In this section we draw “A”, with different size and place within the plane. In Figure 1 we see the character A drawn in different size and place. In Figure 2 we see the same three characters after they have been scaled to the same size and translated to the same area of the plane. Finally in the following set of figures, we can see the plots of the angle, distance and final feature of each “A” character.

It is obvious, that the feature has the same basic structure in all three “A’s”. There are of course some fluctuations, which can be attributed to our hand not being steady while drawing the characters. We can also see a small difference in the length of the features (this will be more obvious in later sections), which is caused by the different speeds of drawing, while the pen is down.

After this section, we will no longer show pictures of the initial characters or their versions after they have been scaled/translated. We will only show pictures of the characters’ features and it can be assumed that the features are extracted by the scaled/translated version of the character, as in Figure 1.

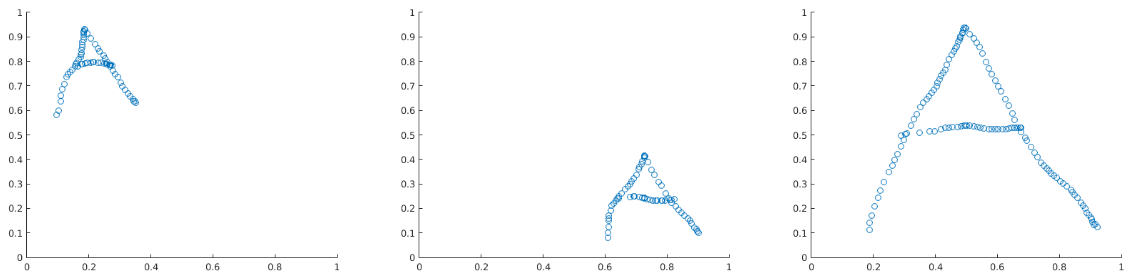


Figure 1: “A” drawn with different size and at a different place

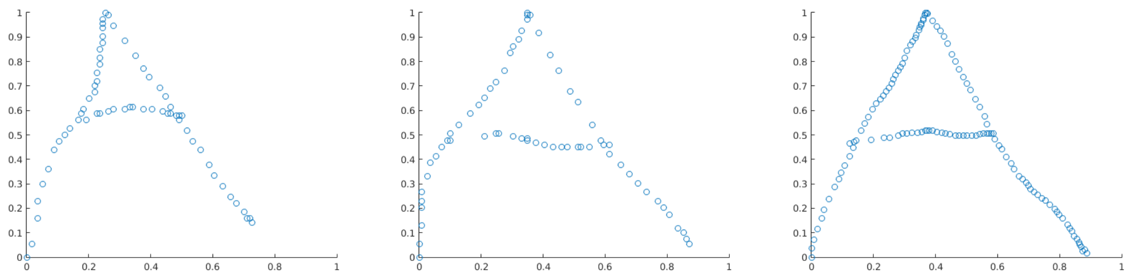
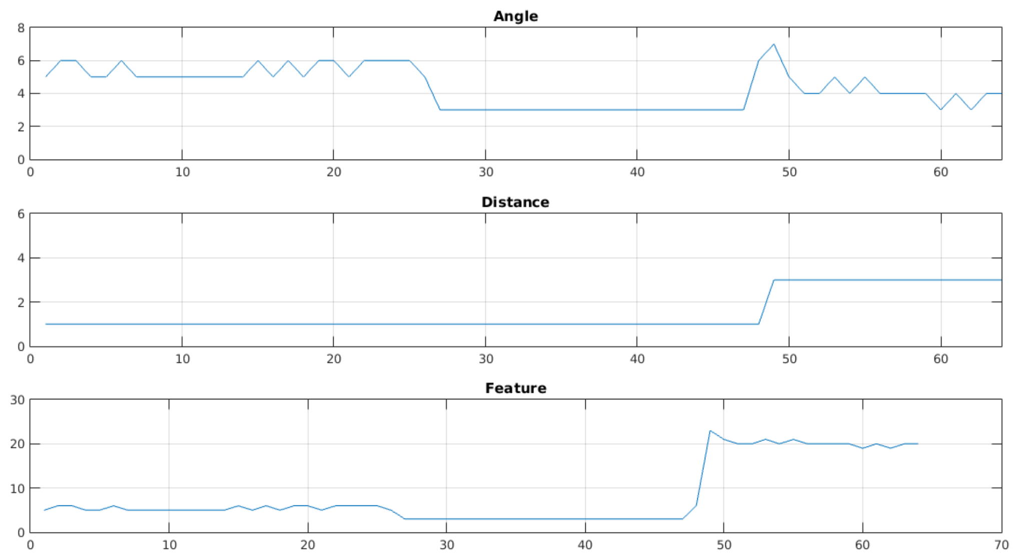
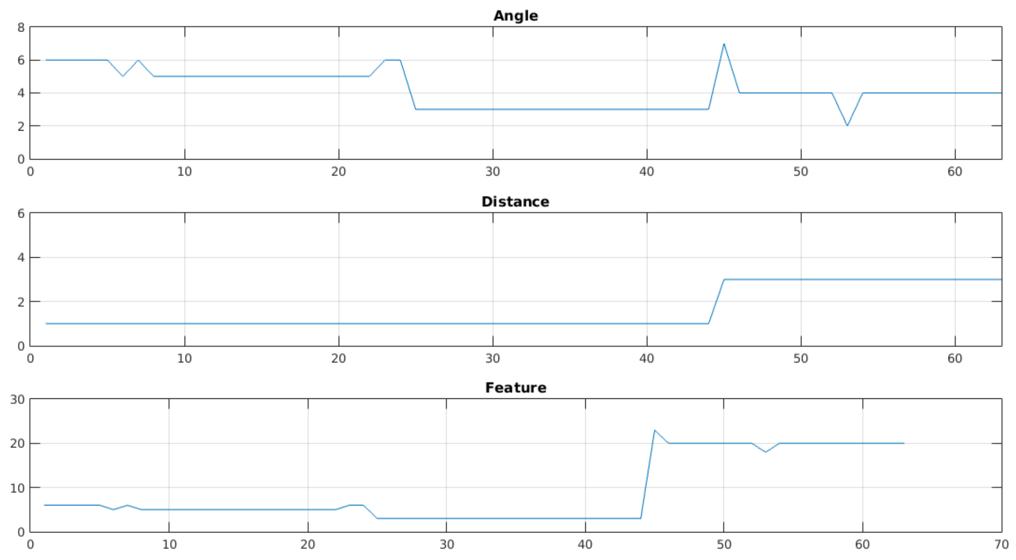


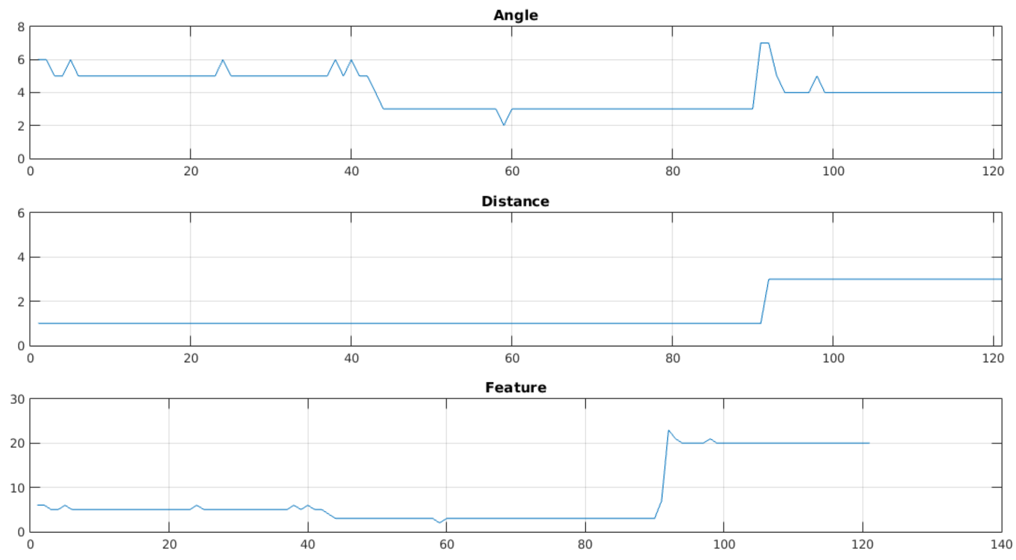
Figure 2: “A’s” of Figure 1 scaled and translated



(a) First “A” Features



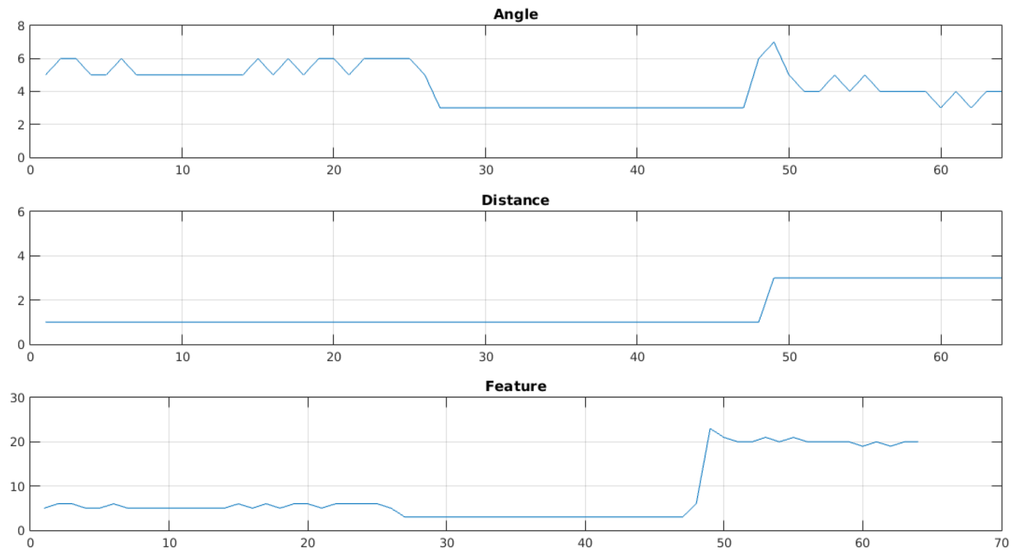
(b) Second “A” Features



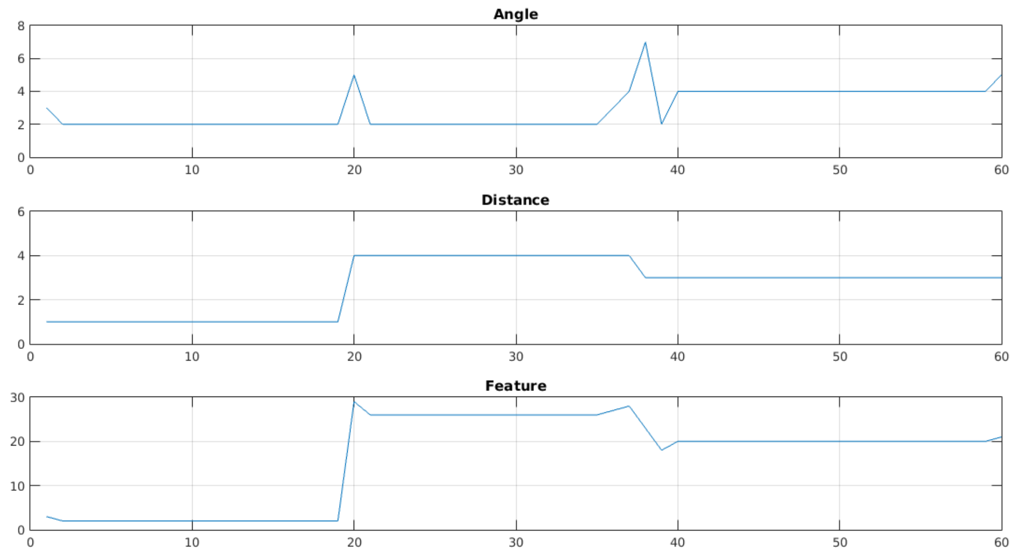
(c) Third “A” Features

Comparison of features for characters "H" and "A"

In this section we draw the characters “A” and “H”. Similarly to the previous section we show the plots for the angle, distance and final feature of these characters.



(a) “A” Features



(b) “H” Features

"A" with wildly different pen movements

This section contains drawing of “A”. While drawing the character, when the pen was in the air, it followed very different paths. It is obvious by the images, that our feature disregards this information and the final result is almost identical.

The difference observed between the features (that looks like they have been scaled over time-frames), is what was mentioned in a previous section. It is caused by the speed of the stroke, while the pen is down, but in no way affected by the movement of the pen, while it’s in the air.

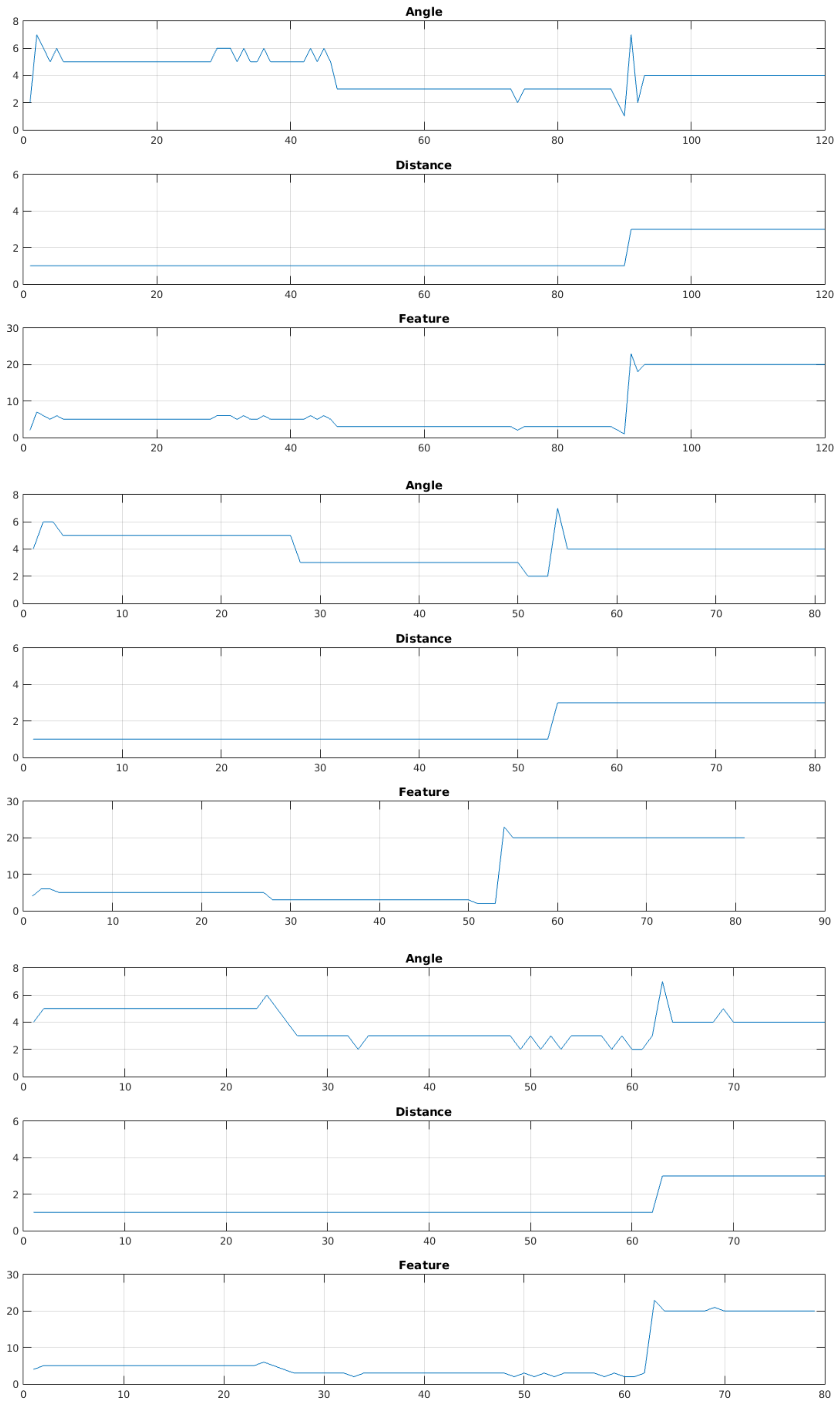
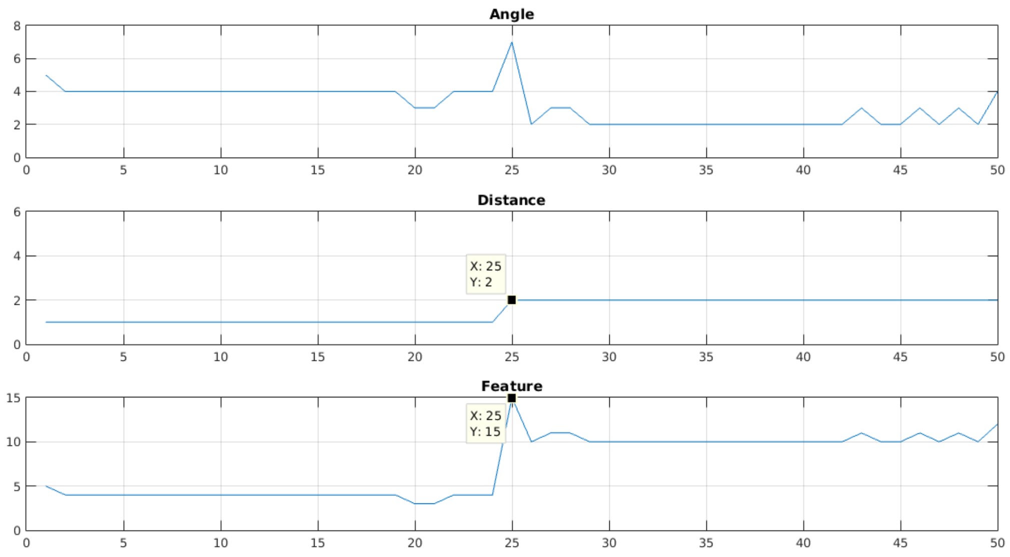


Figure 5: “A” Drawn with wildly different paths for the pen, while in the air

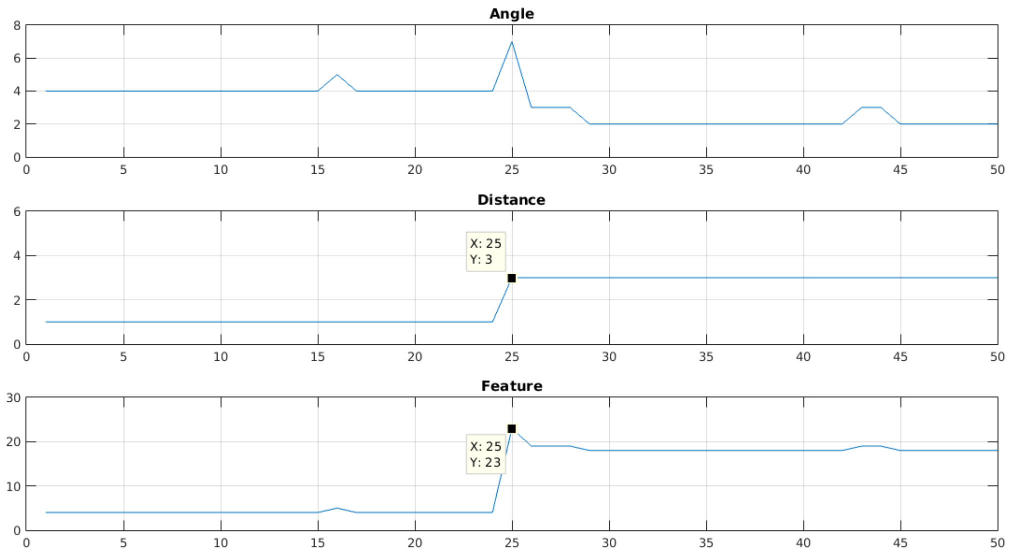
Features for “T” and “+”

When extracting features for “T” and “+”, the angle feature is theoretically identical. For this reason, the feature that differentiates between the two characters is the distance. In the following Figures, we can see the effect that the distance has on the final feature at the moment of the pen’s jump. It is obvious that the final features have similar structure, but different values, which can help a classifier tell them apart.

It has to be noted, however, that, depending on the way of the drawing (which line is drawn first, the horizontal or the vertical?), we may get different jumps between strokes, resulting in different final features, that may actually look more alike for the two characters. This can probably be resolved, by setting a certain “norm” for the drawing of these two characters that will have to be followed by the user.



(a) “T” Character’s Features



(b) “+” Character’s Features

Feature Faults

Since the intermediate features that are used for the construction of the final are screenshots of different timeframes, it is natural that the feature is very much dependent on time.

That is, either the speed of the strokes can affect our feature (longer feature and/or more sparse when strokes are faster), or the sequence of the strokes themselves. The second case is better understood through the following figures.

It is obvious from these figures, that even though we have the same character (disregarding the differences in the width of the character, which is caused by unsteady drawing), the final features are not the same, which would cause trouble at the training of our classifier. To overcome this issue, other than the suggestion made in the previous section (setting a default way of drawing the characters), we can also try to gather a very rich dataset, where we have different instances of the characters, which we can use for the training.

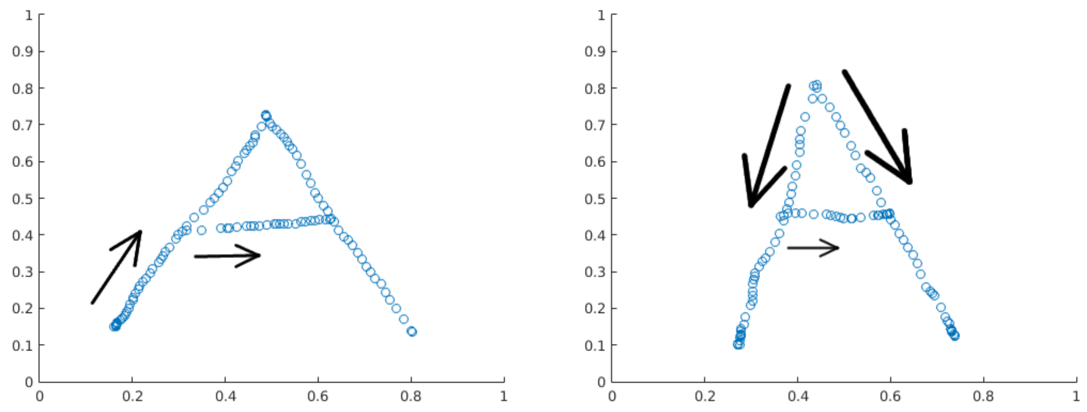


Figure 7: “A” drawn with different strokes

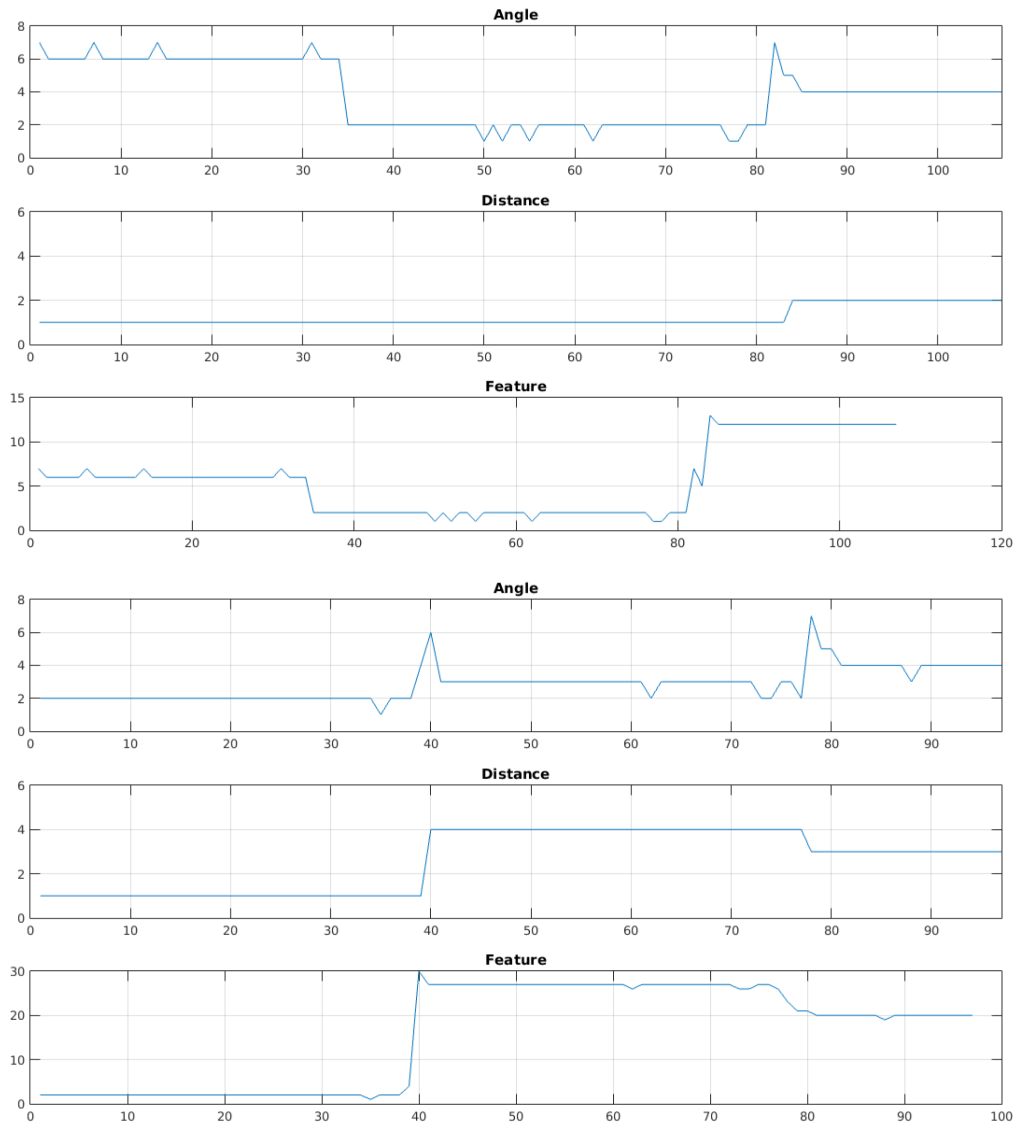


Figure 8: Features for "A's" drawn with different strokes

MATLAB Files

Along with this report we submit a set of MATLAB files, which we explain in this section:

characters.mat This file contains the drawings of the characters used in this report, as well as extra example for all cases. It consists of a number of cell array variables, where each cell contains the output of the function $DrawCharacter()$ for the character drawn for each case described by the name of the cell array.

GetAndPlotFeature.m This function takes as input the sequence of a drawn character as it is return by the function $DrawCharacter()$. It calculates the feature of the character, and produces figures for the initial drawn character, the scaled/translated character and the plots for the angle, distance and final features. Within this function the number of bins for the quantization of the angle and distance is set to 8 and 6 respectively, as mentioned in the first section of the report.

FeatureExtract.m This function is our basic feature extractor. It takes as input a character sequence, like function GetAndPlotFeature(), as well as the number of bins for the quantization procedures. It returns four arrays:

- *sequence:* The scaled/translated character's sequence
- *angles:* The calculated angles' sequence after quantization.
- *distances:* The calculated distances' sequence after quantization.
- *feature:* The final feature of the character.

All the figures included in this report can be reproduced using these functions. It is of course possible, for a new character to be drawn using the function $DrawCharacter()$ and then displayed with the use of the rest of the functions.