CW2 Character Recognition

Introduction

I have attempted to implement simple character recognition on a small subset of characters – namely S, T, and V. To achieve this, I have performed analysis in the Fourier domain, and created a classifier using a small set of training data. I have managed to produce a moderately accurate system, which I have tested on some additional data points. In this report I have detailed the choices I made, and analysed their effects on my classifier and the choices it makes.

Fourier Space Analysis

In order to select features that would form the basis of my classifiers, I began by visually inspecting the log of the Fourier space of the different characters individually. This allowed me to get an overview of how the different shapes in each of the letters related to the output of the Fourier transform.

In an attempt to get more definitive output, I decided to try using Sobel edge detection. When viewed as the raw image output, this appeared to be very effective, and very clearly showed the defining lines of each character. However, when passed through the Fourier transform and viewed in the Fourier domain, the pixelated nature of the output produced very strong horizontal and vertical lines through the origin. Therefore, I decided against using the Sobel operator.

In order to quickly get an idea of how the different variations of each character produced similar patterns in the Fourier domain, I decided to average all the images of each symbol. To achieve this, I read in all images into a 3 dimensional matrix, then averaged this back down to a 2 dimensional image. I then performed the 2D FFT on these averaged S, V, and T inputs. This made it much easier to compare the three different classes side-by-side, incorporating all the variations.

Spectral Feature Extraction

Looking at the three groups of Fourier domains, I saw the three patterns created by the characters were as follows:

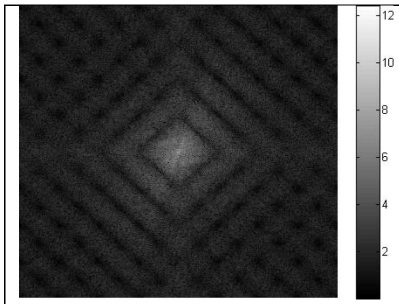	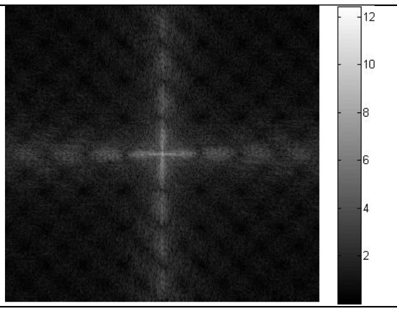	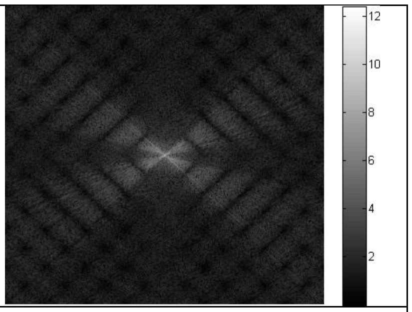
Figure 1 (S): A concentric square rotated 45 degrees, with a fairly regular spread of intensity.	**Figure 2 (T):** Two very strong areas of magnitude along the horizontal and vertical axes.	**Figure 3 (V):** Similar to T, albeit with the intense regions rotated and more diffused.

From looking at these, I chose to use a pair of box features, where I would take the summation of frequency magnitudes in that area of the Fourier domain. I tried two different approaches, shown in **Fig. 4** and **Fig. 5**.

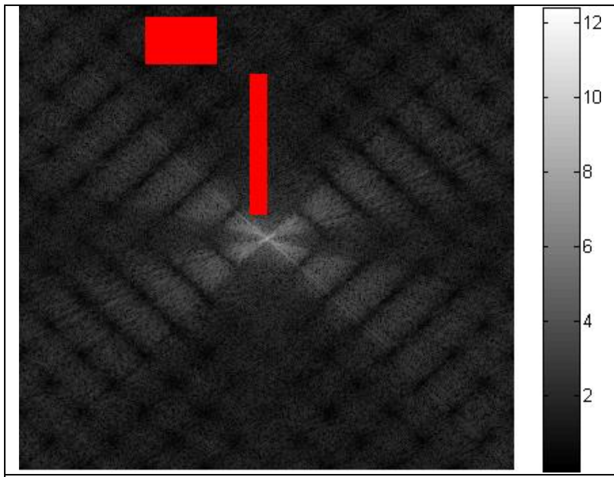	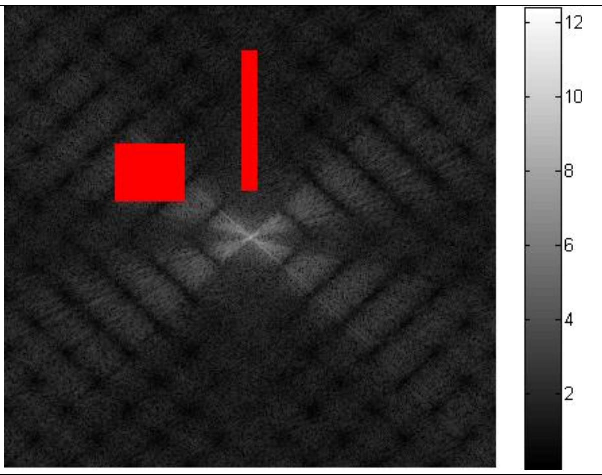
Figure 4: The red rectangles show the box features I chose to differentiate between the three characters, superimposed on the Fourier domain for V. Here I chose to use a vertical bar to capture the strong vertical lines produced by T (corresponding to the horizontal bar). As this was not prominent in the other characters, this feature would identify T. I then chose a rectangle in the lower magnitude space between that bar and the 'spoke' of the V. I planned to use this to highlight the fact that S produced a fairly even spread of intensity, whereas T and V left this area relatively low magnitude.	**Figure 5:** The final features I chose to use. As you can see, I decided to keep the vertical bar (with some adjustments), as that was the most distinctive feature present in the Fourier domain of T. This was very good at distinguishing T from S and V. I however realised after some tests that my use of the 'darker' space to the left was not very effective, as I had overestimated the difference in that area between S and the other symbols. Thus, I tried to capture the diagonal lines of V present in the Fourier domain. This worked quite successfully, so I decided to use these two features moving forward.

Fourier Domain Analysis

I used k-Nearest-Neighbour classification in order to classify character data based on the features I had selected. The first step I took to test the accuracy of my classifier was to use it to classify the data I used to train it. Unless my classifier was particularly bad, this step should be 100% accurate. I had to use a k value of greater than 2 – this is because I was using the nearest point to decide on classification in the event of equal occurrences of two (or more) classes. Therefore, in either case of k = 1 or k =2, I was relying solely on the single nearest neighbour. Intuitively, this would simply return the classification I manually gave each point, as they would be identical values. After some trials of various values, I decided on k = 3. This seemed to produce the most accurate classification, and, being odd, meant there would be no chance of having an equal number of multiple classes.

k-NN Classifier Analysis

To further test my classifier, I drew 2 of each character to use as test data, and passed them through my classifier. (These images can be found in the included archive test_chars.zip.) This resulted in a 100% success rate on my very small test set.

Another method I used to analyse the efficacy of my classifier was to produce a visual representation of the decision boundaries it produced. I achieved this by generating a regular grid of all possible values for my features around the range of magnitudes displayed in the data I had seen so far. I then plotted these coloured in correspondence to their classification by my k-NN classifier, shown in **Fig. 6**.

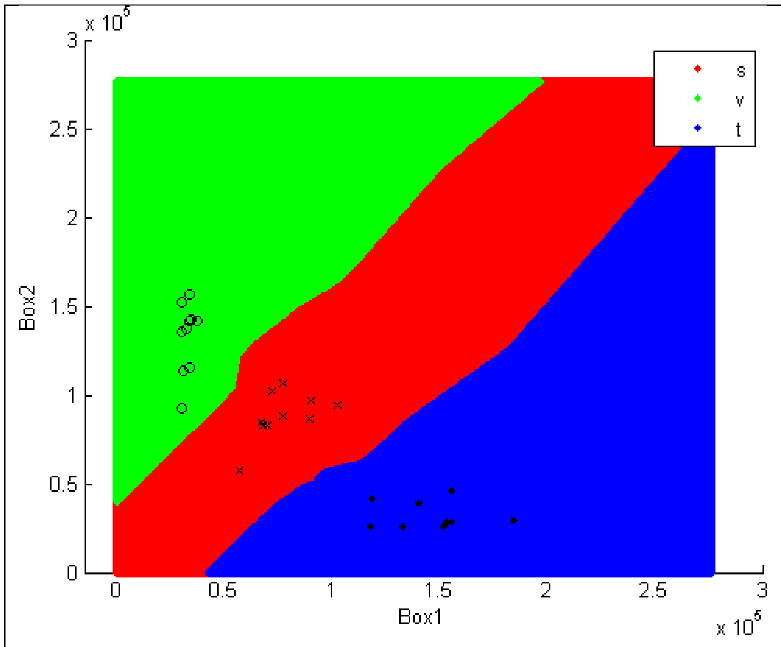

Figure 6: This plot shows the decision boundaries created by my classifier. I have overlaid the scatter plot of the training data.
The x axis represents the sum of absolute magnitude in the box feature that captures diagonals. The y axis represents that of the vertical bar. This clearly explains the pattern shown here – feature box 1 should be high for T, while 2 should be low, and vice versa for V. S occupies the middle band as the magnitude is fairly evenly spread across the Fourier domain.

A & B Classification

In order to demonstrate how my chosen features and classifier performs, I used the classifier on two unknown symbols – A and B. My system classifies both as S characters – so I decided to visually inspect the Fourier domain of these to investigate why that is the case.

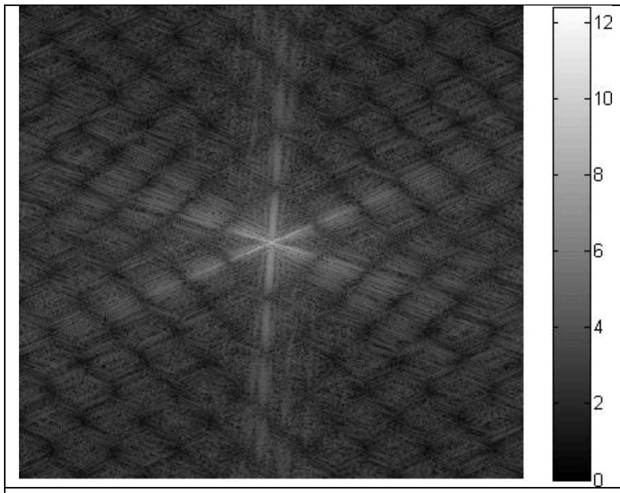

Figure 7: This is the Fourier domain of the A character. As you can see, there is a fairly prominent vertical bar, which is a result of the horizontal bar present in the symbol. This would point to this character being interpreted as T. However, there is also a band of high magnitude in the same region as the feature used to differentiate V characters. (This time, due to the outer shape of the A closely matching that of an inverted V.) As both these areas are relatively high magnitude, this character would fall roughly in the centre of the graph shown in **Fig. 6** – and is thus classed as an S.

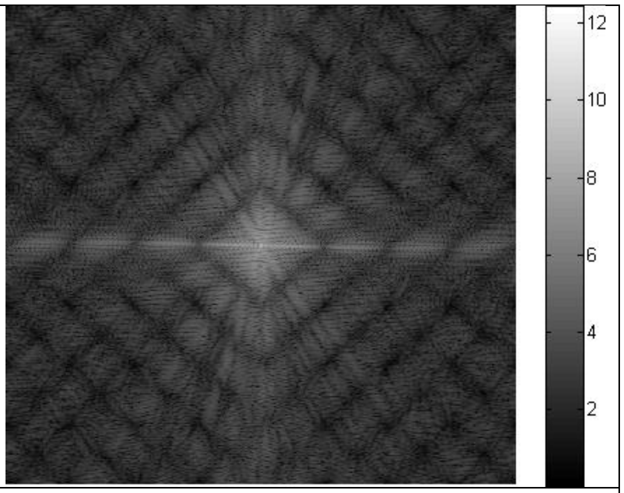

Figure 8: This is the Fourier domain of the B character. In a very similar manner to **Fig. 7**, we can see that the intensities in both of our box features is approximately equal. The most prominent feature of this image is the horizontal band in the centre. Had I chosen my T defining feature box by rotating it 90 degrees around the centre, the B character may have been classified as a T. In my current setup, the relatively small but equal presence of horizontal and diagonal edges in the B symbol, causes a classification of S. (As the sum of absolute magnitudes in both feature boxes will be approximately equal.)

Alternative Classifier

In order to compare and contrast my classifier against a different approach, I decided to implement classification using a decision tree. From inspection of the training points plotted on **Fig. 6**, we can divide the axis using only horizontal and vertical divides and maintain a simple, effective grouping of those points. This tells me that a decision tree could be very effective, and could also, depending on the height of the generated tree, be much faster to classify points.

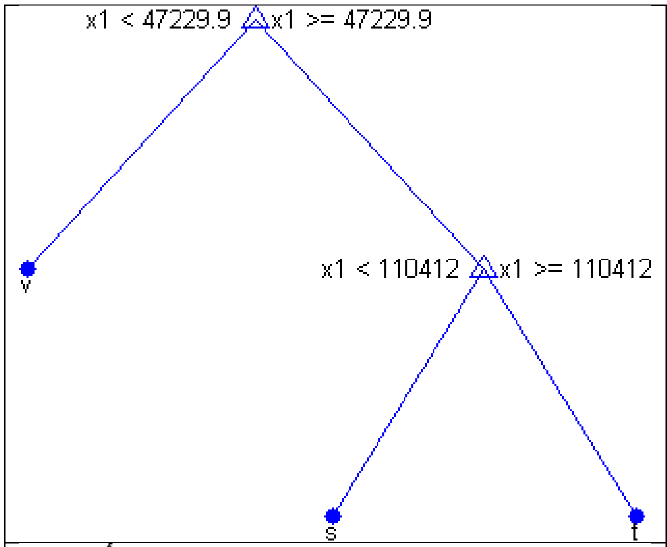

Figure 9: The diagram on the left hand side shows the structure of the decision tree I created. As you can see, the structure of the tree is incredibly simple, thanks to the aforementioned separation between the groups. The algorithm I used to generate the tree has therefore pruned away all decisions based on my second feature, and is relying solely on the first box feature. Because of this, my decision boundaries are very simplistic.

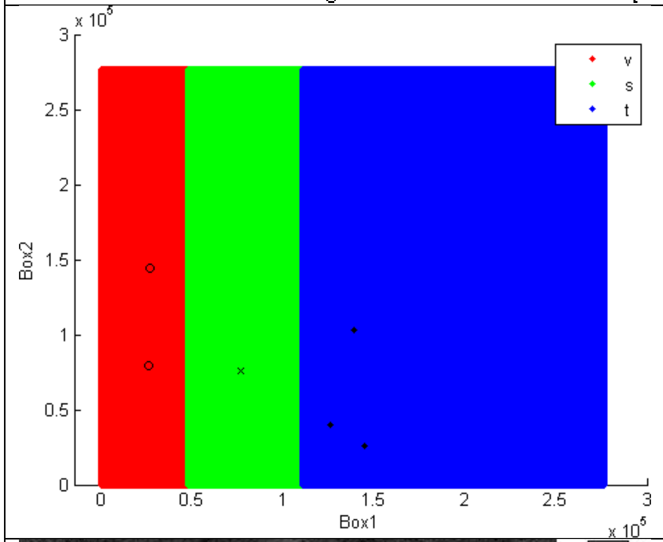

Figure 10: The decision boundaries produced by my decision tree, as detailed in **Fig. 9**. Superimposed on this diagram is a scatter plot of the test points, showing how they are classified by this alternative classifier. Five out of the six symbols are classified correctly, apart from one instance of S which has been identified as T, visible at ~(1.4e5, 1e5). This point is correctly identified by the k-nearest-neighbour classifier detailed earlier, falling within the S band of **Fig. 6**.

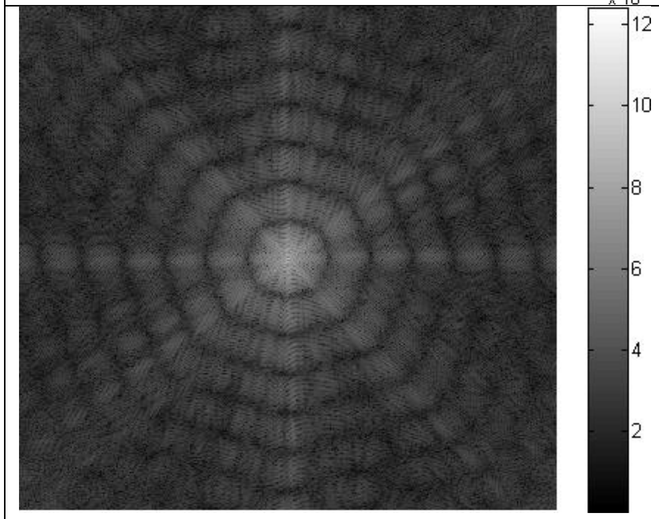

Figure 11: The Fourier domain of the test S that is incorrectly classified by my decision tree. As shown by **Fig. 9**, the reason for this misclassification is due to the tree ignoring feature box 2. In this image, there is a strong area of high magnitude in the area of feature box 1. In the k-NN classifier, the high magnitudes present in the area of feature box 2 would cause this symbol to fall between T and V, and thus be classified correctly. However, as my decision tree is ignoring feature 2, there is nothing to offset the prominence of frequencies in the area of feature 1.

Conclusion

In conclusion, I have designed a simple character recognition system for the letters S, T, and V, which is accurate in classifying a selection of variations of the three characters. I have demonstrated its operation using some unknown symbols, A and B, and explaining their classification by visual inspection of their Fourier domain. I have also contrasted my k-NN classifier with another classifier, provided by a decision tree, which displayed the weakness present in the rather small set of training data used to define the classifier.