Report on the Fourier-analysis character classifier

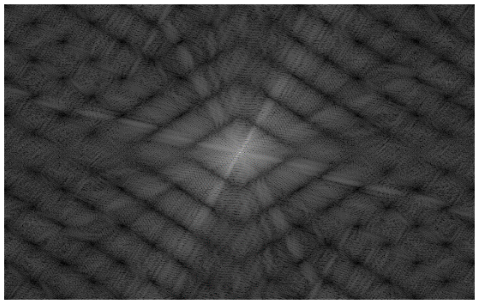
1. Introduction

The task of this assignment was to design and reason about a classifier of manually drawn alphabet characters – e.g. using a touchscreen device or simply a graphics painting software such as MS Paint. More particularly, it asks to implement a means of classifying an input character image as one of three characters – S, V, and T, given a training data set. The aims of this report are to demonstrate my understanding of the task at hand and to explain the results of my work.

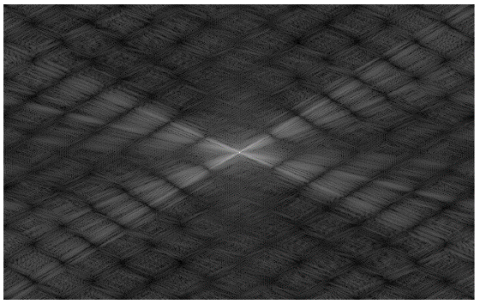
2. Feature Selection & Extraction

The first problem I had to tackle was deciding what features to use to classify characters by means of 2D Fourier space analysis of the training data. There are many different ways of thinking about features and making this choice, which at first made the task seem rather hard.

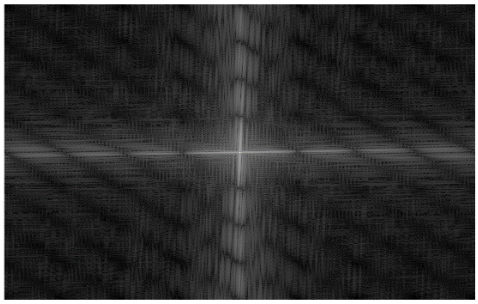
In the case of character recognition in images of the given format, one is looking to detect the presence of edges in particular directions. In the frequency space, these are indicated by the change in frequency in directions orthogonal to those of the edges. What is usually referred to as *magnitude, spectrum,* or sometimes *power spectrum*, represents the magnitudes of different frequency components of an image, transformed from the spatial to the frequency domain. The lowest frequencies are customarily concentrated in the centre. The further away from this point of origin some magnitude is evident, the higher the frequency of the component it belongs to. The spectrum is normally represented on a logarithmic scale for perceptual scaling, since the power can vary by orders of magnitude in an image. This helps isolate periodic structural features and noise in the input image. It is also a common technique to measure the second power in specific regions of the log-scaled magnitude spectrum. In this way one can further accentuate the difference in magnitude at different frequencies to detect linear or textural features.



2D Fourier Transform of "S8.GIF"



2D Fourier Transform of "V3.GIF"

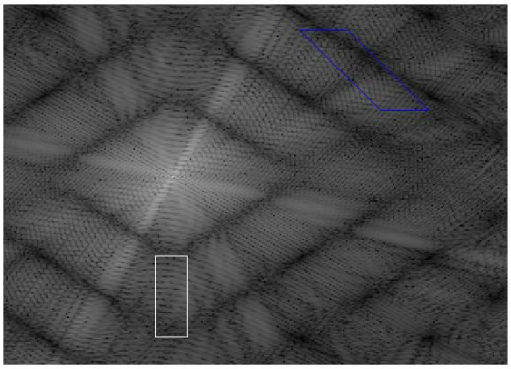
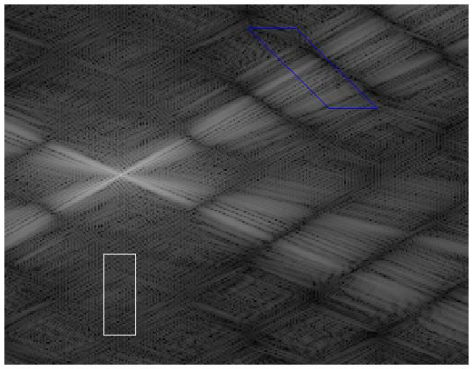
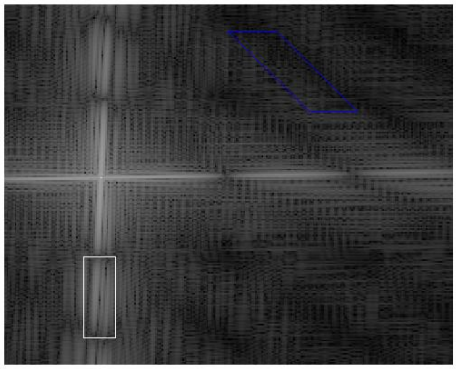


2D Fourier Transform of "T4.GIF"

Thus, I came to the conclusion that it would be best to use power spectrum features from spectral regions. My main consideration became choosing the least possible number of features, that are as informative as possible and preferably small in size, to ensure that my classier is fast, efficient and most importantly – accurate. I soon realised that having features of equal sizes would be very convenient for comparing their powers and plotting them against each-other. I ran a 2D Fast Fourier Transform on the training data and displayed their symmetrically centred magnitude spectrum, on a log scale. I observed the Fourier spaces of multiple training images of each class. It became obvious that they have some quite distinctive features.

The ones of "S" characters are the most complex. Due to the nature of these characters – being comprised of what can be described as two semi-circles – their Fourier Space consists of patterns in multiple directions. The bright lines in the Fourier Space indicate the presence of change of frequency in their direction, hence the orthogonal relation between them and actual lines in the input image. A somewhat ideal digital image of an "S" character consists of lines in all sorts of directions; theoretically - the higher the resolution - the more differently-directed lines. This makes its Fourier Space harder to analyse, as it consists of many linear features to choose from. Therefore I proceeded to examine the Fourier Spaces of images of the other character classes to find features that can make the task of distinguishing them from an "S" character easier.

The "V" and "T" characters are easier to analyse using Fourier space. Since a V consists of two somewhat diagonal lines with an angle between them of usually less than 90^{o} it yields a Fourier space containing bright diagonal lines in both directions. The case with a "T" is even simpler – its Fourier space consists of what looks like two bright lines – one horizontal and one vertical, just like a "T" itself. These two features are completely distinct from each other and also dissimilar enough to the ones in the transform of an "S", which makes them perfect for the purposes of the classifier. As long as there is a relatively good balance between diagonal and horizontal lines in the depiction of an "S" character, the data point, resulting from plotting against each-other the power magnitudes representing these two features in the spectrum, should have a position that enables us to easily classify it as different to both "V" and "T".

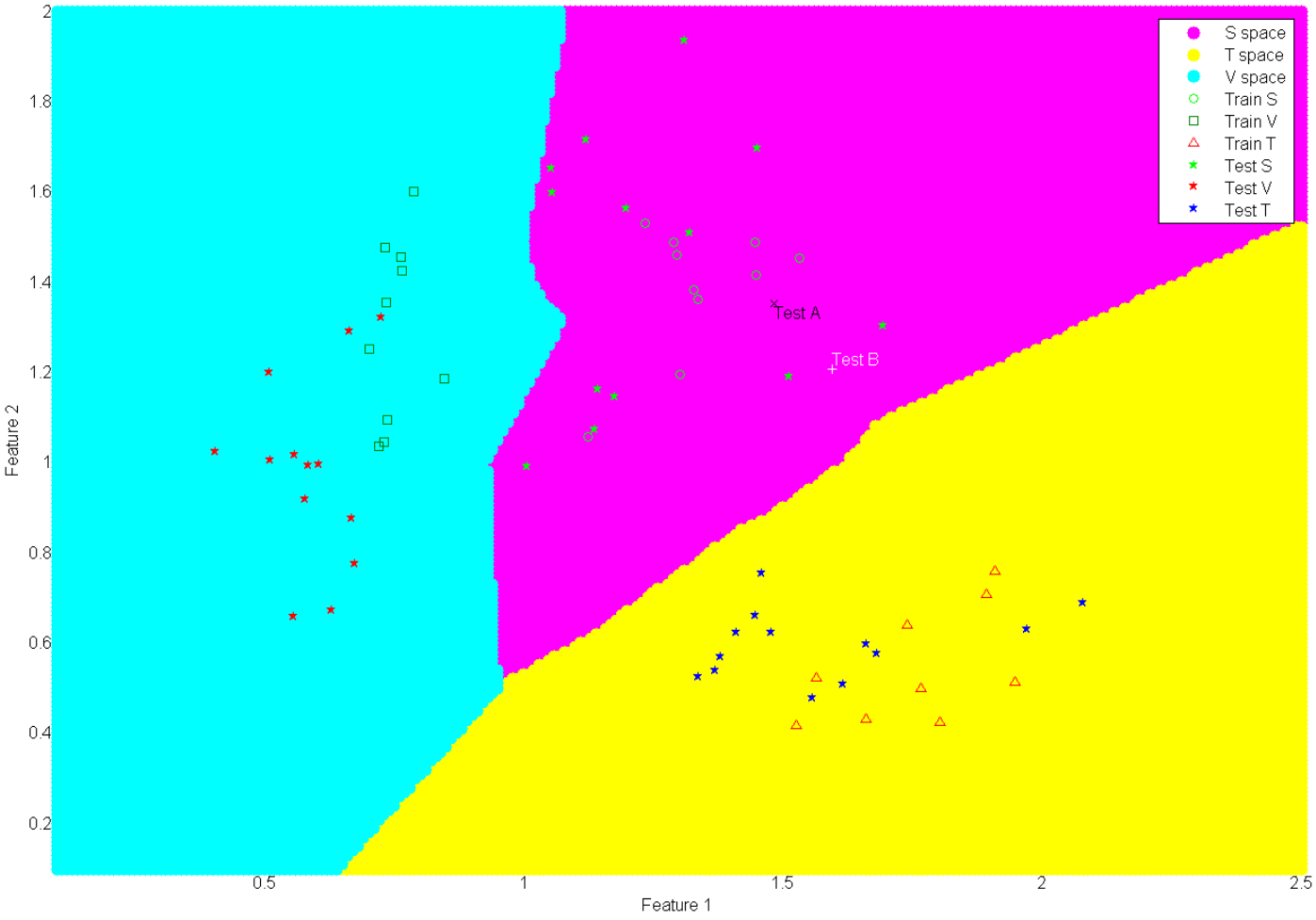




I chose to use a rectangle on the vertical line for my first feature. It is reasonably distanced from the centre of the Fourier space where the similarity between transforms of different images is the greatest due to the concentration of lowest frequencies with high magnitude. For my second feature I decided to use a rhomboid of the same surface size, also positioned at some distance from the centre, which encapsulates a sector of the diagonal lines characterising the transform of a "V" character. As previously mentioned I am using, as numeric features, the summation of the second powers of all the log-scale magnitudes in these regions. I evaluate these for every single training sample, factor them to use smaller numbers for convenience, and then pass them on to the classifier.

3. Classifier & Training data

It was one of the requirements of the assignment to use the nearest-neighbour classification algorithm to classify the features obtained via Fourier space analysis. This classification method requires prior knowledge of the class each training exemplar belongs to, in order to assign a test instance to a class. The instance space is divided into regions containing all points closer to a training exemplar than to any other one, by means of Voronoi tessellation. Each testing instance then falls into the region of its closest neighbour and is classified into its class.

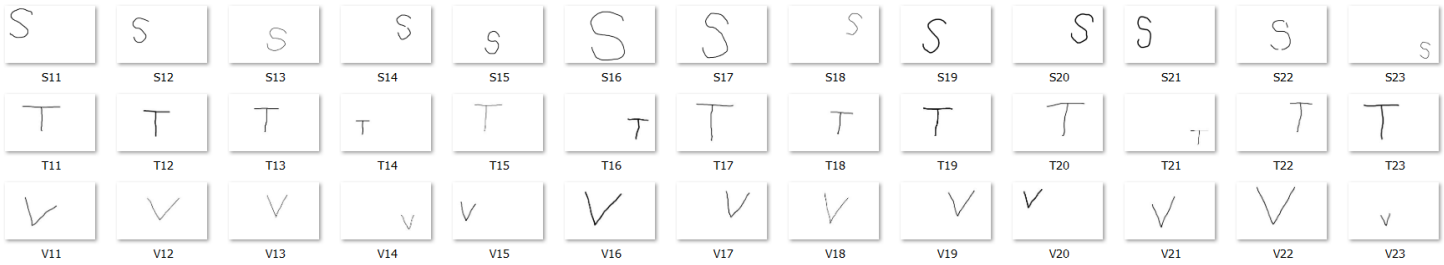
As suggested in the coursework description, I created a regular grid of testing points over a range, containing the coordinates of the training exemplars, defined by the feature values, allowing some extra space for possible offsets. I decided that for the purposes of this classifier it is enough to classify all the points on this grid and colour them differently according to the class assigned by the classifier. So, I passed the exemplar points obtained from the training data set, grouped by their respective class, and the grid of testing instances to a nearest neighbour classifier using 1 nearest neighbour, as required. I then created the coloured plot to visualise the decision boundaries of the classifier. Finally, on the same plot I scattered the training exemplars along with testing instances, also obtained by means of Fourier space feature extraction of training images I generated myself. The results are displayed below. Note that there are only 9 training data points of class T displayed, as two of the exemplars we were given are absolutely identical, namely T2 and T6.



4. Testing the classifier

4.1 Target characters – S, V & T

For the purposes of testing, I manually produced 13 images of each character – S, V and T. I then extracted from their Fourier spaces the same features, as the ones I extracted from the training samples, and plotted them against each other as test data instances on the already classified grid in order to visually represent how the system had classified them. This can be seen on the plot above and the images of the testing characters can be seen below.



As visible from the plot, all testing data points fall into the correct class. Still, some of the "S" characters are very close to the decision boundary. This leaves room for doubts and possibly improvement. Whilst generating the test data I tried to create exemplars with different properties that might challenge the classifier in order to check how it copes with them. I created exemplars of different sizes using brushes with three different thicknesses.

The two testing "S" characters, plotted one above the other, which narrowly miss being misclassified as "V"s for example, are named "S12" and "S14" in the image above. They are both small and contain predominantly diagonal lines. Since the classifier is not using a distinctive feature from the Fourier spaces of "S" samples, but rather relies on them having a combination of the two distinctive features for "V"s and "T"s, to distinguish them it can misclassify an "S" sample containing a high value for one and a low value for the other feature. As described in the feature extraction section the two features represent two small spectrum regions – one on the central vertical line, and one on the top-right diagonal. The use of these features ensures the detection of the presence of a horizontal or a diagonal line in the input image, but in some cases this may not be ideal. On a personal note, maybe choosing a region on the other diagonal could have worked slightly better for distinguishing "S" characters as they are less likely to contain a top-right to bottom-left diagonal-line component. Having said all this I think that the classifier generally works well for a good variety of testing samples.

4.2 Alternative characters - A1 & B1

Testing the system using the "A1.GIF" and "B1.GIF" samples resulted in them being classified as "S" within the decision boundaries of the nearest neighbour classifier. This was rather expected – an "A" character contains both features used by the classifier – the horizontal and the diagonal lines, which means that the resulting data point from plotting these against each-other would be close to the one of an "S", which similarly has high values for both features. A "B" character on the other hand, which consists of two semi-circular components and one vertical line, ought to have a central horizontal line in its Fourier space, and lines and frequency patterns going in all directions. This means that its transformation in the frequency space would be quite similar to the one of an "S", and explains the decision of the classifier.

It is interesting to note that creating a classifier for a greater number of distinct characters – a whole alphabet for example – would require a much more in-depth analysis and more complex feature selection and extraction process. Furthermore, a Fourier analysis character classifier of complexity similar to the one I had to produce, relies on the detection of edges that have specific directions and would be completely ineffective if one has to classify images of characters that are not oriented properly – e.g. same characters, rotated by a significant angle.

Kristian Krastev
kk12742@bristol.ac.uk

5. Using an alternative classifier

Since the decision boundary of the nearest-neighbour classifier I used somewhat resembles a linear one, I decided to examine the use of a nearest-centroid classifier that uses a linear decision boundary.

Here, the centroid of each training data class is evaluated as the mean of all exemplars in it, and can then be used to assign all testing instances, closer to it than to that of any other centroid, to that class. Below is a plot that contains both the outcome of the nearest-neighbour classifier, and the centroids and decision boundary of the nearest neighbour one. Although the nearest-neighbour decision seemed nearly linear, one can see that it is quite different from the nearest-centroid one. In our case there is only one testing instance that has been misclassified, but there is a significant amount of space that is classified differently. The nearest-centroid classifier's decision is highly dependent on the training data exemplars. If they were different, the centroids would be different and hence the decision boundary as well would differ. I assume that the more training data the classifier uses the more accurate the centroid and the decision would be. Still this logically must be the case for any classifier.

Generally the nearest neighbour classifier seems to be the better choice for this task. Its decision boundary is more precise and can be made even smoother by using the extended k-nearest neighbour classifier that uses multiple nearest neighbours to classify a testing instance.