

# Handwritten Signature Recognition From the Ground Up

Tyler Sheffield

## A Starting Place

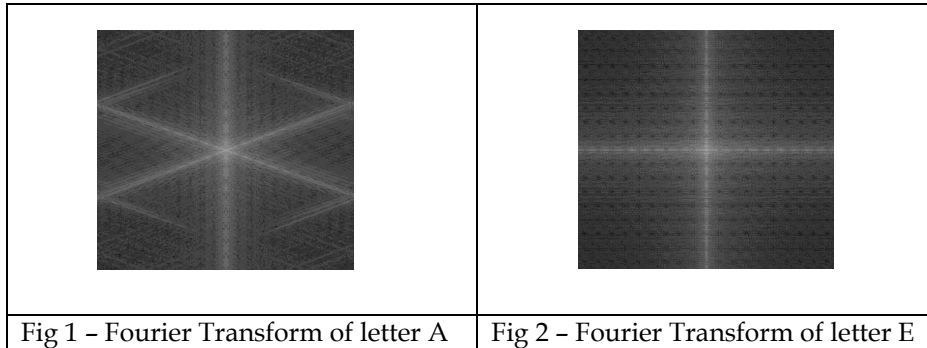
The art of recognizing signatures is very complex. Experts in the field spend years in training in order to comprehend the delicate nuances of signatures. It has been theorized that every signature is distinctive, and for that reason the use of signatures as a biometric has been supported and implemented in various technologies. However, getting a computer to recognize and verify handwritten signatures is no small task. It is, therefore, a widely studied topic; there are countless methods and variations on methods that have been proposed and researched.

Digital image processing is a new field to me. As I began to research this topic, it became apparent that I would first need to understand some of the basics of shape and pattern recognition. In this case the most applicable variety is known as Optical Character Recognition (OCR) which has many uses and is also a widely studied problem. Here, then, is where we will begin.

## A Simple OCR System

The first step in OCR is the pre-processing of the image, which is not the focus of this study. Suffice to say, the steps involved include noise filtering, binarization and segmentation of the image.

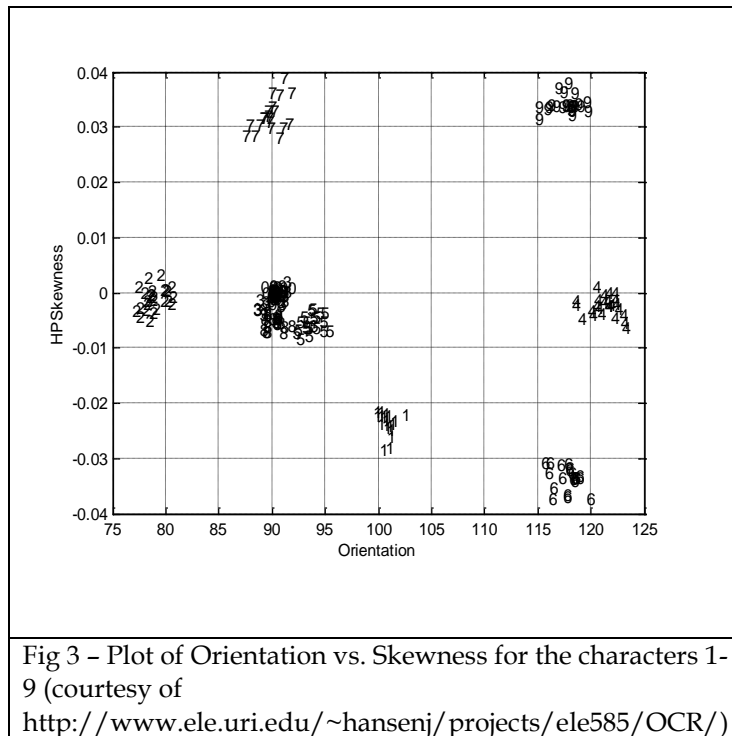
There are a variety of ways to extract features for use in classifying a character. Some methods involve heavy use of the Fourier transform. An example of the uniqueness of these transforms for a couple of characters is shown below (Fig1,2).



My approach is to use geometric features. The following properties can be extracted using the *regionprops* function:

Area	in pixels
Perimeter	in pixels
Centroid	(x,y)
Extent	Area divided by the area of the bounding box (smallest box that fits image inside)
Skewness	third standardized moment
Orientation	angle between x-axis and major axis of ellipse sharing second moment
Eccentricity	distance between foci divided by major axis length

As shown here (Fig 3), these features are useful because different samples of characters cluster together when the values for these features are compared. Obviously, the more features that are used, the better accuracy in the system.



Initially, a five-letter recognition system was created for the letters A, C, E, R, T in a size 48 Arial font. Emphasizing the understanding of basic principles over accuracy, one computer generated training sample for each letter was used to define the system.

Some normalizations and adjustments had to be made. Eventually, the system was able to perfectly identify all 20 letters in a scanned page containing the words RAT, CRATE, EAT, CAR and CREATE.

The system was then expanded to the entire alphabet, in capital letters. The scanned image (Fig 4) was recognized as:

THIG PAGG EQNTAING INZQRWAYIDN RSGARDING THG GPSCIFICATIOMG



Fig 4 - Scanned sentence

The results were promising. Obviously, there are some problem areas with the letters S,G and E among others. A better-trained system would perform much more consistently.

The noise filtering on the scanned image is very limited. The page was crumpled up and again scanned. The OCR system's output was predictably horrific, especially since MATLAB found more characters than there truly were:

YHIG ORGG SRNTRIMG INFHNWZTIOM QEGDRQINF III IICGOGSIVIEATHIINS

I was interested in further exploring the difficulties of the problem and tested the code on computer generated characters with fonts other than Arial. This table (Fig 5) shows the collection of 30 fonts on which the system was tested.

ACERT	ACERT	A C E R T	ACERT	ACERT
<b>ACERT</b>	ACERT	ACERT	ACERT	ACERT
ACERT	ACERT	ACERT	ACERT	<b>ACERT</b>
ACERT	ACERT	ACERT	ACERT	ACERT
A C E R T	ACERT	ACERT	<b>ACERT</b>	ACERT
ACERT	ACERT	<b>ACERT</b>	ACERT	ACERT

Fig 5 - Font table

The results were mixed. As expected, there were serious problems characterizing certain fonts. Others seemed to be fine being compared to features of the Arial font(Fig 6).

<p><b>ACERT → AERRT</b>  A C E R T → T T T T T  <b>ACERT → ACERT</b></p>
----------------------------------------------------------------------------------

Fig 6 - Results of classification of three of the fonts

### Signature Recognition as a Biometric




Signatures are one of the oldest biometrics. Efforts to digitize them and create smart computer systems that can distinguish between forgeries and genuine articles continues today. There are two measures widely used to judge the efficiency of a verification system. FAR (False Acceptance Ratio) is the measure of how often a forgery passes for the real thing. FRR (False Rejection Ratio) is the measure of how often a genuine signature is rejected by the system.

There is a major division between types of verification systems. On-line (dynamic) systems involving real-time digital capture of a signature have an advantage in that they can record multiple features of the signature as a function of time, such as pen pressure and azimuth. Off-line (static) systems rely purely on the signature's image but require less hardware. It is on the latter that this report focuses.

### Signature Feature Extraction


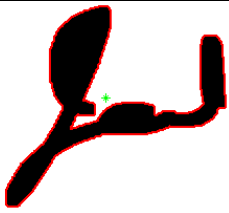
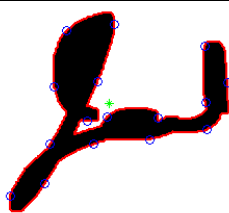
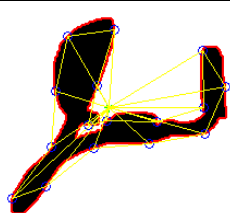
There are a wide variety of methods of signature feature extraction. Some research done on the topic seeks to define the shape of the signature on a point by point basis and use pre-processing techniques like skeletonization (Fig 7,8). Then methods such as line directionality can be

explored, in which each pixel is associated with vector describing connectivity to pixels around it. Others use transforms such as the Hough Transform (Fig 9) which creates  $\rho$  by  $\theta$  matrix (from  $x \cos \theta + y \sin \theta = \rho$ ) by the colinearity detection property. It is very useful in conjunction with tools like *houghpeaks* and *houghlines* and was seriously considered for use on this project.

		
Fig 7 - A digitized signature	Fig 8 - Skeletonized signature	Fig 9 - Hough Transform of signature

However, a proposal by Ferrer, Alonso, and Travieso in “Offline Geometric Parameters for Automatic Signature Verification Using Fixed-Point Arithmetic” uses a unique, computationally less expensive algorithm. This project consists of an implementation and test of a variation on their algorithm.

Consider this signature(Fig 7). We first map the outline, or contour(Fig 11), of a dilated version of the image(Fig 10) and select N evenly-spaced sample points from the outline coordinates (Fig 12). We are interested in  $\Delta r_t$  - the difference in the distances from two adjacent contour points to the centroid of the image;  $\theta_t$  - the angle between a line drawn between the contour point and centroid and the x-axis;  $A_t$  - the number of pixels contained in a triangle drawn between the centroid and each pair of adjacent contour points (Fig 13). Both  $\Delta r_t$  and  $A_t$  are normalized to the largest value.

			
Fig 10 - Dilated version	Fig 11 - Contour mapped and centroid shown	Fig 12 - Contour points sampled	Fig 13 - Triangular masks drawn

Extracting these features gives us an  $N \times 3$  feature vector sequence that characterizes the signature. Now that we have the features, there are a multitude of methods for classifying images by these features.

## Classifying Signature Image Data

Many papers have focused on the usefulness of Hidden Markov Models and neural networks for training systems for signature recognition. The method chosen for this project is string matching.

Using this method, the values of the feature vectors are essentially M-level quantized, with each level being represented by a character. The feature vector sequence will then resolve into three character strings. These can be compared to one another using a function called *strsimilarity* provided by the DIPUM toolbox ([http://www.imageprocessingplace.com/DIPUM/dipum\\_m\\_files/dipum\\_toolbox.htm](http://www.imageprocessingplace.com/DIPUM/dipum_m_files/dipum_toolbox.htm)) which returns a similarity index for two strings on a scale of 0 for completely different strings to infinity for identical strings. A more advanced system would use finite automata for processing language defined by the rules set forth via a grammar. Time constraints limit the development of such a system.

## Proving the Concept and Testing

For these test, the signature image data is borrowed from a database compiled by Ferrer, Alonso, and Travieso for use in their research and made available for use at (<http://www.gpds.ulpgc.es/download/>). Each signature is identified by a number. The database also includes multiple versions of forgeries of each genuine signature.

For proving the concept of the system, let us choose  $N = 16$  contour samples and  $M = 8$  levels of quantization. We will train the system with five samples,  $S=5$ , each of eight distinct signatures.

First of all, we will establish the accuracy of the system for distinguishing between signatures. We do this by classifying eight samples of genuine signatures for each of the eight distinct signatures and create this confusion matrix (Fig 14).

	Sig 10	Sig 11	Sig 13	Sig 20	Sig 21	Sig 23	Sig 24	Sig 30
Sig 10	5		2		1			
Sig 11		8						
Sig 13			8					
Sig 20				6	2			
Sig 21	1				7			
Sig 23						6	2	
Sig 24				3			5	
Sig 30						1		7

Fig 14 - Signature confusion matrix

We will now estimate the system efficiency in the realm of verification by classifying a series of eight genuine signatures and eight forgeries. The results are shown below (Fig 15).

<b>Forgeries</b>	Accepted	Rejected	<b>Real</b>	Accepted	Rejected
Sig 10	0	8	Sig 10	4	4
Sig 11	2	6	Sig 11	5	3
Sig 13	1	7	Sig 13	6	2
Sig 20	5	3	Sig 20	6	2
Sig 21	3	5	Sig 21	6	2
Sig 23	3	5	Sig 23	5	3
Sig 24	3	5	Sig 24	4	4
Sig 30	0	8	Sig 30	6	2
	FAR	0.265625		FRR	0.34375

Fig 15 - FAR and FRR results for N=16, M=8, S=5

It is desirable to increase the efficiency of this system, and we can do so by increasing to ten the number of training samples used. Although this lowers the FAR as expected, the tradeoff is that the FRR rises (Fig 16)

<b>Forgeries</b>	Accepted	Rejected	<b>Real</b>	Accepted	Rejected
Sig 10	0	8	Sig 10	2	6
Sig 11	2	6	Sig 11	4	4
Sig 13	2	6	Sig 13	5	3
Sig 20	1	7	Sig 20	0	8
Sig 21	6	2	Sig 21	6	2
Sig 23	1	7	Sig 23	2	6
Sig 24	2	6	Sig 24	4	4
Sig 30	0	8	Sig 30	5	3
	FAR	0.21875		FRR	0.5625

Fig 16 - FAR and FRR results for N=16, M=8, S=10

Again we seek to improve the system. Let us now increase Ns to 32 and increase M to 16 (Fig 17).

<b>Forgeries</b>	Accepted	Rejected	<b>Real</b>	Accepted	Rejected
Sig 10	1	7	Sig 10	2	6
Sig 11	1	7	Sig 11	3	5
Sig 13	0	8	Sig 13	1	7
Sig 20	0	8	Sig 20	4	4
Sig 21	1	7	Sig 21	1	7
Sig 23	0	8	Sig 23	4	4
Sig 24	0	8	Sig 24	0	8
Sig 30	2	6	Sig 30	1	7
	FAR	0.078125		FRR	0.75

Fig 17 - FAR and FRR results for N=32, M=16, S=10

## Results Revisited

Although a lower FAR is far more critical than a lower FRR, this system is certainly not optimal. We should be able to lower the FRR to below 20%, while keeping the FAR very low. There are a couple of ways to approach this that will be discussed, but an implementation of an optimal system is beyond the objective of this project.

As mentioned earlier, the method used to extract the feature vector sequence for these signatures is a variation on that proposed by Ferrer, Alonso, and Travieso. In their paper they also included a second type of Cartesian coordinate-based extraction, also producing a three column feature vector sequence. Including these features, or perhaps others such as Hough lines, would add to the accuracy of the system.

Another important aspect to consider in the creation of the strings for string matching is that the feature values are uniformly quantized. A better system would involve analysis of where the feature values are most likely to lie, and fixing a corresponding non-uniform quantization (quantization with expansion, for example) to better differentiate the values. We could also consider doing away with string matching entirely, and focusing on a feedback-oriented training system such as a neural net. This would have been the preferred approach given more time to expand the project.

The problem of signature recognition/verification is not an easy one, and the volume of research work dedicated to techniques designed to improve existing methods will attest to that. It is extremely difficult to create a highly effective, selectively discriminative system. There exist many more reliable biometrics; even so, the number of challenging geometric and intelligence puzzles in this particular field, and its ancient origins, ensure that handwritten signature recognition will be actively studied and improved upon in the years to come.



## References

---

- Jesse Hansen, "A Matlab Project in Optical Character Recognition (OCR)", <http://www.ele.uri.edu/~hansenj/projects/ele585/OCR/>
- Leszynski Group, Inc., "Digital Ink Signatures - Concepts and Technologies", <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dntablet/html/tbconinksig.asp>
- Richard Wisneski, "Digital Image Processing: Optical Character Recognition (OCR) and JSTOR", <http://www.personal.kent.edu/~rwisnesk/jstor/jstor.htm>
- Miguel A. Ferrer, Jesu's B. Alonso, and Carlos M. Travieso, "Offline Geometric Parameters for Automatic Signature Verification Using Fixed-Point Arithmetic", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 27, NO. 6, JUNE 2005
- E. N. Zois, A. A. Nassiopoulou, V. Anastassopoulos, "Signature Verification Based On Line Directionality", Research Development Telecommunications Laboratory, Electronics Department, Athens Technological Educational Institution
- Author Unknown, "Signature Verification", [www.cse.msu.edu/~cse802/Papers/802\\_Signature\\_Verification.pdf](http://www.cse.msu.edu/~cse802/Papers/802_Signature_Verification.pdf)

## Implementation Notes

---

The implementation discussed in this paper was done in MATLAB. The code (m-files) is submitted electronically in a .zip file. Following is documentation touching on each of the files written for this project. This information can also be found in README.doc in the submitted .zip file.

<b>OCR Related Files</b>	
cc.m	Performs KNN classification for five characters
cc_full.m	Performs KNN classification for full alphabet
mf.m	Simple median filter
seg.m	Uses library tools to segment and extract features from a sample set of five characters
seg_full.m	Uses library tools to segment and extract features from a sample set of the full alphabet
thresh.m	Simple binarization
train.m	Extracts data from the training set of five characters
train_full.m	Extracts data from the training set of the full alphabet

<b>Signature Related Files</b>	
geofunc.m	Performs dilation and distance/angle/pixel feature extraction on signature image
sigclass.m	Imports a forged or genuine signature image from the database, and calls geofunc and smatch on the image in order to classify it
sigtrain.m	Imports a given number of signature images from the database training set, and calls geofunc and smatch on those images
skel.m	Skeletonizes signature and executes a Hough transform on the result
smatch.m	Performs 8-level quantization into character strings and string matching for $N_s \times 3$ feature vector sequence
smatch16.m	Performs 16-level quantization into character strings and string matching for $N_s \times 3$ feature vector sequence