Handwritten Gregg Shorthand Recognition

R. Rajasekaran
Principal
F X Polytechnic College
Tharuvai, Tirunelveli

K. Ramar
Principal
Einstein College of Engineering
Tirunelveli

ABSTRACT

Gregg shorthand is a form of stenography that was invented by John Robert Gregg in 1888. Like cursive longhand, it is completely based on elliptical figures and lines that bisect them. Gregg shorthand is the most popular form of pen stenography in the United States and its Spanish adaptation is fairly popular in Latin America. With the invention of dictation machines, shorthand machines, and the practice of executives writing their own letters on their personal computers, the use of shorthand has gradually declined in the business and reporting world. However, Gregg shorthand is still in use today.

The need to process documents on paper by computer has led to an area of research that may be referred to as Document Image Understanding (DIU).The goal of DIU system is to convert a raster image representation of a document. For Example, A hand written Gregg shorthand character or word is converted into an appropriate Symbolic Character in a computer (It may be a scanned character or online written character).Thus it involves many disciplines of computer science including image processing, pattern recognition, natural language processing, artificial intelligence, neural networks and database system. The ultimate goal of this paper is to recognize hand written Gregg shorthand character and Gregg shorthand word.

General Terms

Pattern Recognition, Character Recognition, Shorthand script Recognition. Artificial Neural Networks

Keywords

Gregg Shorthand Recognition, Competitive Artificial Neural Network, Shorthand Script Recognition

1. INTRODUCTION

Gregg shorthand is easy to learn, read and write. Gregg Shorthand may be learned in from one-third to one-half the time required by the old systems. The records made by its writers prove this beyond all questions. Gregg Shorthand is the most legible shorthand in existence. In the public shorthand speed contests, writers of the system have established the highest official world's records [1] for accuracy of transcripts on difficult matter. These records were made in competition with experienced reporters who used the older systems, and in contests conducted by reporters and teachers who wrote such systems. Manifestly, the insertion of the vowels, the absence of shading, the elimination of position-writing, and the elimination of the minute distinctions of form, all contribute to legibility. The easy, natural appearance of the writing in Gregg Shorthand appeals to every impartial investigator. The absence of distinctions between light and heavy characters, the continuous run of the writing along one line, as in longhand, instead of constant changes of position now on the line, then above the line, and then, perhaps, *through* or *below* the line will be noticed at first glance. Next, the investigator will probably attribute much of the natural, pleasing appearance of the writing to that uniform slant of the writing with which both hand and eye are familiar. Only those who have had previous experience with shorthand, however, will be able to appreciate fully how much elimination of numerous dots and dashes minute marks that have to be placed with great precision alongside the strokes contributes to fluent writing. The simple and logical writing basis of Gregg Shorthand enables a writer of it to use it in any language with which he or she is familiar. Special adaptations of the system have been published for Spanish, French, German, Italian, Portuguese, Polish, Gaelic, and Esperanto. Adaptations to other languages are in preparation. The Spanish adaptation of the system is used in more than 300 schools in Spanish-speaking countries, and there is a quarterly magazine devoted to it. With the invention of modern computers and hand held devices the usage of such wonderful Gregg shorthand is declined.

Leedham & Downtown [2],[3],[4]and [5] and Hemanth Kumar [7] have addressed the problem of automatic recognition of PSL strokes and techniques like Hough transformation and regional decomposition have been discussed.

Nagabhusan, Basavaraj Anami and Murali [6],[7],[8],[9],[10],and[11] have addressed many different issues on knowledge based approach for handwritten pitman shorthand language strokes. Rahul kala, Harsh Vazirani, Anupam Shukla and Ritu Tiwari [12] addressed offline character recognition using Genetic Algorithm. Shashank Araokar [13] explained handwritten alphabet recognition. A very less work is available for Gregg shorthand.

We consider that it is necessary to convert the handwritten Gregg shorthand character or word in to its equivalent English word or character. In this paper, A Gregg character or Gregg word can be recognised and converted into its equivalent English character or word.

In this paper a single shorthand character or word drawn into the text area can be digitalized and it can be learned or recognized. If learning mode is selected the computer learns the character, if it is a recognition mode then the computer Compares the drawn character with already stored patterns using CANN(Competitive Artificial Neural Networks) and displays the result in probabilistic method as well as characteristic method. There are 24 main characters and 19000 frequently used words in Gregg shorthand. In this paper various types of Gregg shorthand characters and words

are tested and the results were presented. Our future proposal is to recognise and convert all those 19000 frequently used Gregg shorthand words. Recognition rate for online and offline Gregg shorthand was compared. Efficiency of offline is better over online.

The rest of the paper is organized as follows: Section (2) focuses on Artificial Neural networks and Section (3) emphasizes on HGSR (Handwritten Gregg Shorthand Recognition) algorithm for Gregg shorthand recognition and Section (4) for the Experimental results and conclusion.

2. ARTIFICIAL NEURAL NETWORKS

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as neurons. Each of these neurons can connect with up to 200,000 other neurons, although 1,000 to 10,000 are typical.

The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them. It also comes from genetic programming and learning. The individual neurons are complicated. They have a myriad of parts, sub-systems, and control mechanisms. They convey information via a host of electrochemical pathways. There are over one hundred different classes of neurons, depending on the classification method used. Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. In short, it is nothing like the currently available electronic computers, or even artificial neural networks. These artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism. They do it in a primitive way. But for the software engineer who is trying to solve problems, neural computing was never about replicating human brains. It is about machines and a new way to solve problems.

On the contrary, neural network researchers are seeking an understanding of nature's capabilities for which people can engineer solutions to problems that have not been solved by traditional computing.

To do this, the basic unit of neural networks, the artificial neurons, simulates the four basic functions of natural neurons. Figure 1 shows a fundamental representation of an artificial neuron.

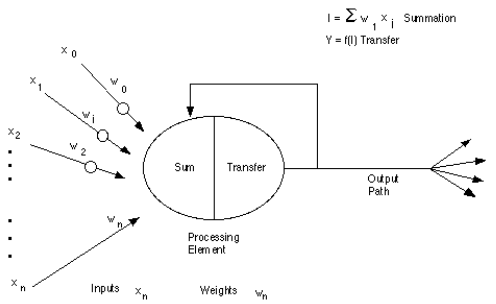


Figure 1 A Basic Artificial Neuron.

In Figure 1, various inputs to the network are represented by the mathematical symbol, x(n). Each of these inputs are multiplied by a connection weight. These weights are represented by w(n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output. This process lends itself to physical implementation on a large scale in a small package. This electronic implementation is still possible with other network structures which utilize different summing functions as well as different transfer functions.

Some applications require "black and white," or binary, answers. These applications include the recognition of text, the identification of speech, and the image deciphering of scenes. These applications are required to turn real-world inputs into discrete values. These potential values are limited to some known set, like the ASCII characters or the most common 50,000 English words. Because of this limitation of output options, these applications don't always utilize networks composed of neurons that simply sum up, and thereby smooth, inputs. These networks may utilize the binary properties of ORing and ANDing of inputs. These functions, and many others, can be built into the summation and transfer functions of a network.

Other networks work on problems where the resolutions are not just one of several known values. These networks need to be capable of an infinite number of responses. Applications of this type include the "intelligence" behind robotic movements. This "intelligence" processes inputs and then creates outputs which actually cause some device to move. That movement can span an infinite number of very precise motions. These networks do indeed want to smooth their inputs which, due to limitations of sensors, come in non-continuous bursts, say thirty times a second. To do that, they might accept these inputs, sum that data, and then produce an output by, for example, applying a hyperbolic tangent as a transfer functions. In this manner, output values from the network are continuous and satisfy more real world interfaces.

Other applications might simply sum and compare to a threshold, thereby producing one of two possible outputs, a zero or a one. Other functions scale the outputs to match the application, such as the values minus one and one. Some functions even integrate the input data over time, creating time-dependent networks.

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins.

There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full blown sense of being truly self learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network.

This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adoption. At the present time, unsupervised learning is not well understood. This adoption to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments.

This paper illustrates both supervised and unsupervised learning mechanisms of Gregg shorthand recognition.

3. HGSR ALGORITHM

3.1 Existing System

The recognition of characters from scanned images of documents has been a problem that has received much attention in the fields of image processing, pattern recognition and artificial intelligence.

Classical methods in pattern recognition do not as such suffice for the recognition of visual characters due to the following reasons:

1. The ‘same’ characters differ in sizes, shapes and styles from person to person and even from time to time with the same person.
2. Like any image, visual characters are subject to spoilage due to noise.
3. There are no hard-and-fast rules that define the appearance of a visual character. Hence rules need to be heuristically deduced from samples.

As such, the human system of vision is excellent in the sense of the following qualities:

1. The human brain is adaptive to minor changes and errors in visual patterns. Thus we are able to read the handwritings of many people despite different styles of writing.
2. The human vision system learns from experience: Hence we are able to grasp newer styles and scripts with amazingly high speed.
3. The human vision system is immune to most variations of size, aspect ratio, color, location and orientation of visual characters.

In contrast to limitations of classical computing, Artificial Neural Networks (ANNs), that were first developed in the mid 1900’s serve for the emulation of human thinking in computation to a meager, yet appreciable extent. Of the several fields wherein they have been applied, *humanoid computing* in general and *pattern recognition* in particular have been of increasing activity. The recognition of visual (optical) characters is a problem of relatively amenable complexity when compared with greater challenges such as recognition of human faces. ANNs have enjoyed considerable success in this area due to their humanoid qualities such as adapting to changes and learning from prior experience. So we have decided to take this ANN for this paper.

The strategy used for recognition can be broadly classified into three categories, namely: structural, statistical and hybrid. Structural techniques use some qualitative measurements as features. They employ different methods such as rule-based, graph-theoretic and heuristic methods for classification. Statistical techniques use some quantitative measurements as features and an appropriate statistical method for recognition. In hybrid approach, these two techniques are combined at appropriate stages for representation of characters and utilizing them for recognition. Depending on the problem, anyone of these techniques can be utilized while accommodating the variations in handwriting.

3.2 Description of Algorithm

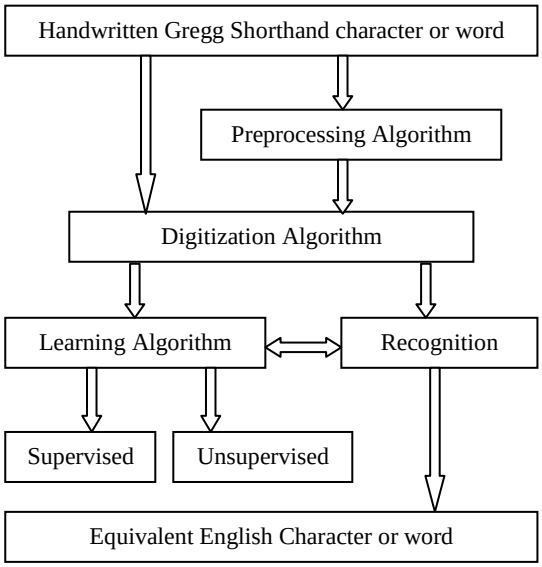


Figure 2 Block Diagram of HGSR Algorithm

Block diagram of Handwritten Gregg Shorthand Recognition Algorithm is as above. Handwritten Gregg shorthand image was taken as an input (offline or online).Preprocessing algorithm performs all the filtering; Normalization operations. Digitization algorithm performs character digitization operation. Learning algorithm performs learning of new Gregg character or word either in supervised mode or unsupervised mode. Validation algorithm compares the input Gregg character with already trained set of Gregg characters or words and produced the result possibilities in percentage. This HGSR algorithm is a combination of all the above sub algorithms.

a) Preprocessing Algorithm

Gregg shorthand Image is given as an input to this algorithm. It performs filtering operations, normalizing operations, resizing operations etc. The output image can be saved and given as an input to Digitizing algorithm.

Begin preprocess

Step1: The Gregg character offline is taken as an input (Image)

Step2: Image size is determined

Step3: Filtering can be applied based on the need of the image either it can be applied for softening, or sharpening or embossing, normalizing etc.

Step4: Save the image
End preprocess

b) Digitization algorithm

Preprocessed image or a normal Gregg shorthand image can be given as an input to this algorithm. Output of this algorithm is given as an input to Recognition or Learning Algorithm.

Begin digitize
Step1: The Gregg offline /online character is given as an input and the size of the grid is selected based on the need of accuracy.
Step2: Image is raster scanned block by block and identifies the left border
Step3: Scans the image and identifies the right border
Step4: Scans the image and identifies the top border
Step5: Scans the image and identifies the bottom border
Step6: Calculate width and Height of the image
Step7: Calculate whether the drawn Gregg character is tall and skinny or short and fat and the location where the shorthand image is available in the text area.
Step8: Scans the image from the detected boundary from left, to right
Step9: if block1 (black) then fix Grid1 (black)
Step10: Repeat step9 until the entire boundary scans complete
Step11: Get the digitized image of the drawn (or offline) shorthand character
End digitize

The following Figure 3(a) and (b) shows the output of digitization algorithm.

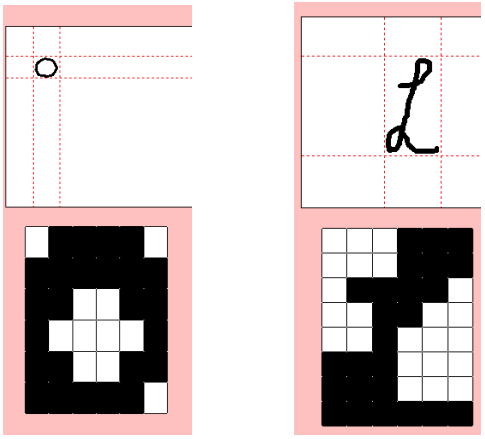


Figure 3 (a) (b)

(a) Offline Gregg Character 'a' and its digitization

(b) Online Gregg word 'aback' and its digitization

In the digitization process, the input image is sampled into a binary window which forms the input to the recognition system. In the above figure, the alphabet 'a' has been digitized into *6X8=48* digital cells, each having a single color, either black or white. It becomes important for us to encode this information in a form meaningful to a computer. For this, we assign a value *+1* to each black pixel and *0* to each white pixel and create the binary image matrix *I* which is shown in the figure 4.

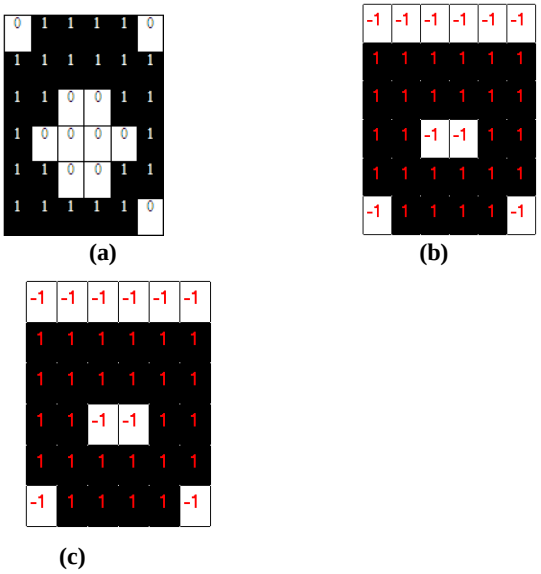


Figure 4.a) Binary digitized input Matrix I

b. Gregg matrix G

c) Learning Algorithm

This algorithm is used to train the neural network either in supervised mode or unsupervised mode. In supervised training, both the inputs and the outputs are provided. In Unsupervised Training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data.

Begin learning

Step1: New character or word to be learned is taken as an Input
Step2: if learning (supervised)

Begin supervised
Step1: Add the new character name into the neural net File
Step2: Save the name of the new character into the Neural net
Step3: learning is applied to the new character in the Neural net
Step4: Repeat step3 until learning completes.
Step5: The network then processes the inputs and Compares its resulting output against the desired output.
Step6: Learning result is stored into the neural net file.
End supervised

Step3: if learning (unsupervised)

Begin unsupervised
Step1: create limit percentage is given as an input
Step2: If the best recognition is below the create limit percentage then a new character will be created.
Step3: learn limit percentage is given as an input
Step4: If the best recognition is above that percentage then learning will take place on that character.
Step5: Repeat step4 until learning completes.
Step6: Learning result is stored into the neural net file.
End unsupervised
Step4: Repeat step2, step3 until learning completes.

End learning

d) Recognition algorithm

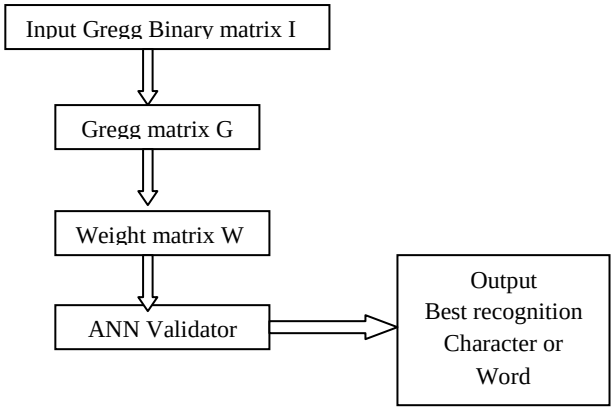


Figure 5 Block diagram of recognition algorithm

This algorithm is used to recognize the Gregg character

Begin Recognition

Step1: Digitized Gregg character matrix 'I' is given as an input
Step2: Scan each and every element of the matrix I
Step3: Create a matrix G then copy all the elements of I and replace the value '0' with '-1'
If I (i, j) =1Then G(i, j) =1
Else:
If I (i, j) = 0 Then G(i, j) = -1
Step4: The input matrix *G* is now fed as input to the neural network.
Step5: It is typical for any neural network to learn in a supervised or unsupervised manner by adjusting its *weights*.
Step6: For the *kth* character to be taught to the network, the weight matrix is formed and denoted by *Wk*.
Step7: At the commencement of teaching (supervised training), this weight matrix is initialized to zero.
Step8: Whenever a character is to be taught to the network, an input pattern representing that character is submitted to the network.
Step9: The network is then instructed to identify this pattern as, say, the *kth* character in a knowledge base of characters.
Step10: The weight matrix *Wk* is updated in the following manner: Wk(i,j)=Wk(i,j)+G(i,j)
Step11: The weight matrix Wk is given as an input to ANN Validator
Step12: ANN Validator compares the weight matrix with already available neural network character set.
Step13: If a link between an input neuron and an output neuron is positive, that means that if the input is on then the total score for that output neuron is increased by a small amount.
Step14: If the link is negative, then it follows that if that input is on, the corresponding output has its score reduced by an amount.
Step15: The output neuron with the highest score (and thus the best match) is considered the winner.
Step16: The output was produced to all the possible Probabilistic percentage of all the character available in the character net

3.3 Performance Analysis

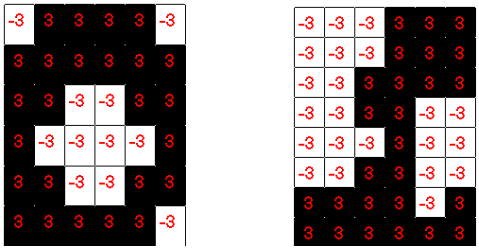


Figure 7 a. Weight Matrix W for Character 'a'
b.Weight Matrix W for Character 'aback'

Figure. 7 a and b gives the weight matrix, say, '*Wa*' corresponding to the Gregg alphabet 'a'. and the corresponding weight matrix 'Waback' for the Gregg word 'aback' The matrix has been updated thrice to learn the alphabet 'a' so the value is 3 and -3. It should be noted that this matrix is specific to the alphabet 'a' alone. Other characters shall each have a corresponding weight matrix.

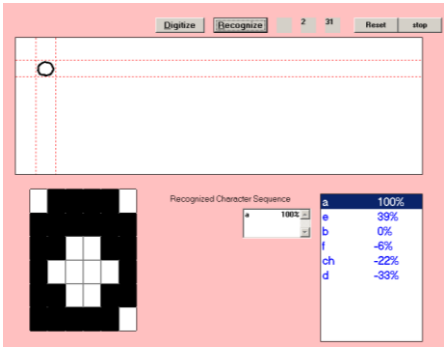


Figure 8 Recognition of Gregg character 'a'

In the above figure 8 the adaptive performance of the network can easily be tested by an example: Gregg character 'a', and by giving repeated learning to the neural network we can get up to 100% recognition result.

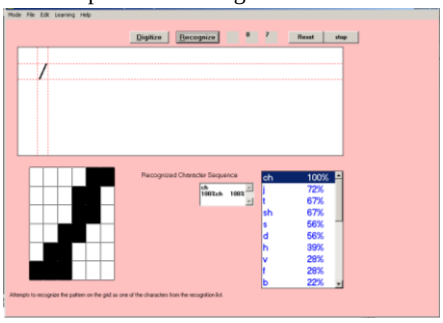


Figure 9 Recognition of Gregg character 'ch'

The neural system has some direct advantages that become apparent at this stage:

1. The method is highly adaptive; recognition is tolerant to minor errors and changes in patterns.
2. The knowledge base of the system can be modified by teaching it newer characters or teaching different variants of earlier characters.
3. The system is highly general and is invariant to size and aspect ratio.
4. The system can be made user specific: User-profiles of characters can be maintained, and the system can be made to recognize them as per the orientation of the user.

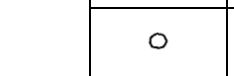
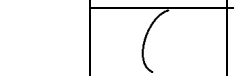
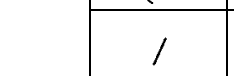
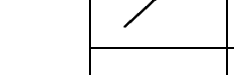
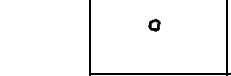
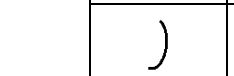
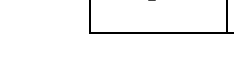
The dimensions of the input matrix need to be adjusted for performance. Greater dimensions, higher the resolution and better the recognition. This however Increase the time-complexity of the system which can be a sensitive issue with slower computers. Typically, *32X32* matrices have been empirically found sufficient for the recognition of Gregg handwritten characters. For Intricate scripts, greater resolution of the matrices is required.

As already illustrated in the previous example, efficient supervised teaching is essential for the proper performance. Neural expert systems are therefore typically used where human-centered training is preferred against rigid and inflexible system-rules.

4. EXPERIMENT RESULTS

24 Gregg alphabet characters and 3 sample frequently used words were taken for this experiment.

TABLE 1.Comparision Between Recognition Of Offline & Online Gregg Alphabet Characters

GREGG ALPHABET SYMBOL	ENGLISH ALPHABET	OFFLINE RECOGNITION TIME IN ms	RECOGNITION QUOTIENT IN %	ONLINE RECOGNITION TIME IN ms	RECOGNITION QUOTIENT IN %
	a	7	100	6	61
	b	5	100	5	83
	ch	5	100	5	72
	d	7	100	8	72
	e	6	100	7	61
	f	6	100	6	83
	g	6	100	6	94
	h	7	100	7	67
	i	6	100	7	61
	j	6	100	6	78
	k	4	100	6	100

Findings:-

1 Recognition time of Gregg characters or Gregg word for both online or offline is more or less same.
2 Algorithm gives 100% accuracy for offline characters or words
3 Algorithm gives an average result of 73% recognition quotient Q accuracy for online characters or words.
4 If the user drawn character size varies Recognition quotient Q for online characters also varies.
5 Online recognition accuracy may come to 100% by giving proper learning either supervised or unsupervised to the network.

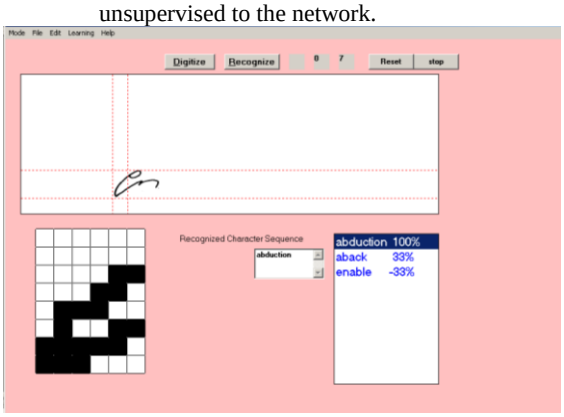


Figure 10 Recognition of Gregg word "abduction"

Table2.Comparision Between Recognition Of Offline & Online Gregg Frequently Used Words

GREGG ALPHABET SYMBOL	ENGLISH WORD	OFFLINE RECOGNITION TIME	RECOGNITION QUOTIENT IN %	ONLINE RECOGNITION TIME	RECOGNITION QUOTIENT IN %
	abduction	7	100	6	79
	enable	6	100	6	78
	aback	8	100	9	64

A simplistic approach for recognition of Gregg characters using artificial neural networks has been described. The advantages of neural computing over classical methods have been outlined. Despite the computational complexity involved, artificial neural networks offer several advantages in pattern recognition & classification in the sense of emulating adaptive human intelligence to a small extent. Our Future work is to recognise rest of 19000 frequently used words of Gregg shorthand.

5. REFERENCES

[1] About Gregg shorthand's history and world records http://gregg.angelfishy.net/anaboutg.shtml

[2] Leedham C G, Downtown A C 1984 On-line recognition of short forms in Pitmans' handwritten shorthand. *Proc. 7th Int. Conf. on Pattern Recognition*, Montreal, pp 2.1058–2.1060

[3] Leedham C G, Downtown A C 1986 On-line recognition of Pitmans' handwritten shorthand – An evaluation potential. *Int. J. Man-Machine Studies* 24: 375–393

[4] Leedham C G, Downtown A C 1987 Automatic recognition and transcription of Pitman's handwritten shorthand - An approach to short forms. *Pattern Recogn.* 20: 341–348

[5] Leedham C G, Downtown A C 1990 Automatic recognition and transcription of Pitman's handwritten shorthand. *Computer processing of handwriting* (eds) R Plamondon, C G Leedham (Singapore:World Scientific)

[6] Leedham C G, Downtown A C, Brooks C P, Newell A F 1984 On-line acquisition of Pitman's handwritten shorthand as a means of rapid data entry. Proc. Int. Conf. on Human-Computer Interaction,

[7] Hemanth Kumar G 1998 *Automation of text production from Pitman shorthand notes*. Ph D thesis,University of Mysore, Mysore

[8] Nagabhushan P, Anami B S 1999 A knowledge based approach for composing English text fromphonetic text documented through Pitman shorthand language. *Int. Conf. On Computer Science (ICCS-99)*, New Delhi, pp 318–327

[9] Nagabhushan P, Anami B S 2000 A knowledge based approach for recognition of grammalogues and punctuation symbols useful in automatic English text generation from Pitman shorthand language documents. Proc. Natl. Conf. on Recent Trends in Advanced Computing (NCRTAC-2000),Thirunelveli, pp 175–183

[10] Nagabhushan P, Murli 2000 Tangent feature values and cornarity index to recognise handwritten PSL words. *Proc. Natl. Conf. on Document Analysis and Recognition (NCDAR)*, Mandya, India,pp 49–56

[11] Nagabhushan P, Anami A knowledge-based approach for recognition of handwritten Pitman shorthand language strokes

[12] Rahul kala, Harsh Vazirani, Anupam Shukla and Ritu Tiwari offline character recognition using Genitic Algorithm IJCSI International Journal of Computer Science Issues, March 2010 pp 16-25

[13] Shashank Araokar Visual Character Recognition using Artificial Neural Networks

[14] Anil K. Jain, Jianchang Mao, K. M. Mohiuddin, Artificial Neural Networks: A Tutorial, Computer, v.29 n.3, p.31-44, March 1996

[15] Simon Haykin, Neural Networks: A comprehensive foundation, 2nd Edition, Prentice Hall, 1998

[16] Alexander J. Faaborg, Using Neural Networks to Create an Adaptive Character Recognition System, March 2002, available at: http://web.media.mit.edu/~faabor/research/ cornell/hci_neuralnet work_finalPaper.pdf

[17] E. W. Brown, Character Recognition by Feature Point Extraction, unpublished paper authored at Northeastern University, 1992, available at: http://www.ccs.neu.edu/home/feneric/charrecnn.html

[18] Dori, Dov, and Alfred Bruckstein, ed. Shape, Structure and Pattern Recognition. New Jersey: World Scientific Publishing Co., 1995.

[19] Gorsky, N.D. "Off-line Recognition of Bad Quality Handwritten Words Using Prototypes." Fundamentals in Handwriting Recognition. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[20] Impedovo, Sebastiano. "Frontiers in Handwriting Recognition." Fundamentals in Handwriting Recognition. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[21] Licolinet, Eric, and Olivier Baret. "Cursive Word Recognition: Methods and Strategies." Fundamentals in Handwriting Recognition". Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[22] Simon, J.C. "On the Robustness of Recognition of Degraded Line Images." Fundamentals in Handwriting Recognition". Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[23] Wang, Patrick Shen-Pei. "Learning, Representation, Understanding and Recognition of Words - An Intelligent Approach." Fundamentals in Handwriting Recognition. Ed. Sebastiano Impedovo. New-York: Springer-Verlag, 1994.

[24]Yaeger, Larry S., Brandyn J. Webb, and Richard F. Lyon. "Combining Neural Networks and Context-Driven Search for Online, Printed Handwriting Recognition in the Newton." A.I. Magazine. 19(1): 73-89, 1998 Spring.

[25]Young, Tzay Y., and King-Sun Fu, ed. Handbook of Pattern Recognition and Image Processing. New York: Academic Press, Inc., 1996.