2011 International Conference on Document Analysis and Recognition

A Compression Scheme for Handwritten Patterns Based on Curve Fitting

Kamal Gupta, Manish Bansal, Santanu Chaudhury
Department of Electrical Engineering
Indian Institute of Technology, Delhi
New Delhi, India
Email: {kamalgupta308, manishbansal.iitd, schaudhury}@gmail.com

***Abstract*— We present here an idea of compression of user fed data from a touch screen input interface for storage and transmission over relatively lower bandwidth. The input is taken in the form of hand-written text, graphics, symbols or patterns and recorded as strokes in order of their temporal occurrence. The patterns are segmented into primitive forms each of which is then modeled with third order B-Spline Curves. The number of control points driving the Spline Curve is determined beforehand by recognizing the dominant points in the pattern. The significant reduction of redundancy in data can be exploited in wide application base including low-cost handheld device communication. This algorithm hence proposes a language independent tool for recording the handwriting of user in its original essence. Results obtained from preliminary testing on MATLAB and Android platform show significant improvement in compression ratio over the traditional storage and compression schemes.**

Keywords- Handwriting, Compression, B-Spline Curves

I. INTRODUCTION

Handwriting is the most basic tool used for expression of thought which may consist of use of different languages or diagrams. Besides conveying the syntactic meaning, handwriting also reflects one's behaviour. With the upcoming technologies in the field of software, the whole world is going digital. However with more than 5000 languages in the world, it's not possible to develop software support for all of them. With the availability of touch pads and smart phones we expect huge proliferation of handwritten information. Also with various limitations and language dependence of handwriting recognition schemes, it becomes mandatory to develop efficient techniques to directly encode the hand-drawn information digitally.

In this paper, we have proposed a scheme for compression of online hand-written data. High orders of compression are achieved by approximating the patterns using B-Spline Curves. This can be used in context of applications like communication tool in handheld mobile phones, chat interfaces, and even as an instrument of making this world paperless, hence serving the environment.

There has been consistent work on online and offline handwriting recognition in the past. Various issues like connecting cursive handwriting strokes, character recognition and modelling etc. have been addressed in [7, 10, 11, 12, 13, 14, and 16]. Most of these algorithms are robust but have limited usability in terms of language dependency and classification accuracy. Liu in [6] presents a model for lossy compression using delta modulation and [8] first trains a Markov Model based recognizer offline, followed by online approximations using arcs of ellipse.

However in all these works, the focus is on modeling the handwriting of a person and trying to regenerate the cursive characters in particular scripts to replace the ASCII characters. To the best of our knowledge, there has been no focus on a generic compression scheme for hand drawn patterns. The novelty of our work is to model any pattern in any language or form, and regenerate the information as was input earlier. Exploiting high redundancy and temporal correlation in patterns, our scheme achieves remarkable compression which is not focused in previous works. The issue of high costs of transmission bandwidth is a major concern in developing countries of Africa, Asia and Latin America. Our technique shall be highly successful in such a customer base because of lesser requirements in terms of bandwidth (hence cost effective). Moreover, there is deep penetration of mobile services in these nations and the communication usually takes place in their native scripts which complies with our scheme.

Our concept of the use of B-Splines for curve approximation is inspired from [3] in which there has been attempt of 3D reconstruction of opaque microscopic objects like grains by approximating various sections using Non Uniform Rational B-Splines (NURBS). We have generalised the concept for any curve that may as well be hand drawn, given the temporal occurrence of information. Also we apply handwriting heuristics to pre-process the patterns and use uniform B-Splines instead which generate equally acceptable results.

The rest of this paper is organized as follows: Section II provides an overview of the coding and decoding algorithm. In Section III we discuss the proposed B-Spline Curve fitting scheme. The results, practical applications and conclusions are discussed in Section IV to VI.

II. OVERVIEW OF ENCODING AND DECODING SCHEME FOR COMPRESSION

In this section, we discuss a top-level overview of the compression scheme. In any typical scheme strokes of hand-drawn patterns are stored as very closely occurring points.

But there is high degree of redundancy in such representation of data, e.g. a simple line segment (as in character 'I' or hyphen '-') can be represented by its starting and ending coordinates only. So a given stroke can be broken into temporal windows that consist of either straight lines [5] or regions which are more profoundly curved and cannot be represented by a set of straight lines (Fig. 1.(a)).

The curve segments are approximated with polynomials. We find that B-Splines are highly efficient for this and they have been discussed in detail in next section. The final compressed data is then entropy encoded using Huffman scheme and can be decoded at the receiving end to regenerate the patterns. A brief layout is shown in Fig. 2.

A. *Removing sharp kinks from the input pattern*

In a stroke (pen-down till pen-up), user may draw any complex figure that may include some sharp corners. We observed that at such a sharp edge or kink, the hand movement is slower and more than one data point is recorded at the same location. So using heuristics, wherever a cluster of points is found in a stroke, it is divided into two.

B. *Breaking Pattern into temporal windows*

The input data are stored in form of closely occurring points *(P_i)* with known coordinates *(X_i, Y_i)* with respect to origin (say the bottom left corner of input panel/screen). To break the given pattern into the above described temporal windows, following steps are followed:

- For all points in the given stroke, angle α_i between segments joining points P_i and P_{i-1}, and points P_i and P_{i+1} is calculated (Fig. 1.(b)).

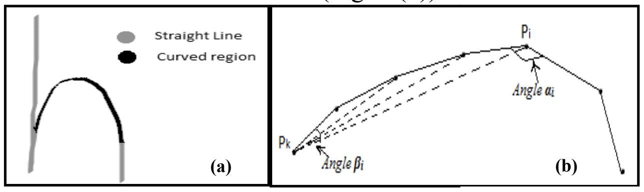
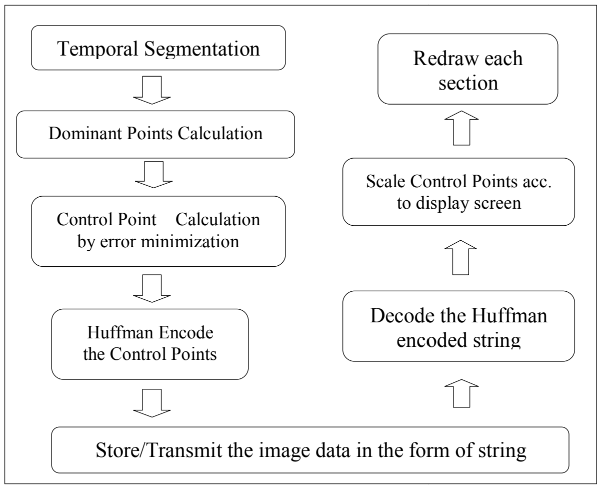


Figure 1. (a) Character 'h' divided into regions of lines and curves. (b) Dividing set of data points into straight lines and curves



Temporal Segmentation
Dominant Points Calculation
Control Point Calculation by error minimization
Huffman Encode the Control Points
Store/Transmit the image data in the form of string
Decode the Huffman encoded string
Scale Control Points acc. to display screen
Redraw each section

Figure 2. Layout of encoding and decoding scheme

- From a reference point (the first point of stroke segment) P_k, angle β_i between segments joining points P_k and P_{k+1}, and points P_k and P_i is calculated (Fig. 1.(b)).
- α_i and β_i are scanned with increasing ***i***. If both are less than a preset threshold value then points from P_k to P_i are considered lying on same *line segment*.
- If β_i exceeds the threshold, the current *line segment* is terminated; *k* is updated to current *i* and the process is repeated over remaining stroke segment.
- If α_i exceeds the threshold, the current sequence being highly curved cannot be modelled with line segments. Sequence is terminated at P_{i-1} and the points henceforth are tagged as *curve segment* till α_i less than the threshold is encountered.

C. *Approximating curve segments*

The *curve segments* can be approximated using cubic polynomials. Since a curve segment can be quite complex and might not be representable with a single cubic polynomial, we break the *curve segment* into smaller subsets. The segment is broken at points where slope tends to infinity (first derivative exceeds a threshold) and double derivative approaches zero.

To represent the curve using cubic polynomial we simply need to store the four coefficients of powers of x and x-coordinates of the starting and ending points. The above described method reduces number of points to some extent. However as the curve becomes more complex, the sequences of line segments and curves get smaller hence reducing the degree of compression. Moreover issues like end point continuity of higher order are not addressed. The error increases further on rounding off coefficients. Hence we approach more powerful curve representations (B-Splines discussed next) than simple cubic polynomials.

D. *Huffman encoding final string*

The final string to be sent consists only of digits 0-9 and symbols used as delimiters. It is completely wasteful to use standard 8 bit ASCII character encoding. Besides, statistically some symbols have higher probability of occurrence. Hence we efficiently employ Entropy (Huffman) encoding [9] with number of bits assigned to symbols proportional to their occurrence frequency.

E. *Decoding*

At receiving end, the string is decoded using the Huffman dictionary. Various segments are then reconstructed by generating closely occurring points of the curve they belong to with the help of their respective coefficients.

III. USING RATIONAL B-SPLINE CURVES

Splines are special functions defined piecewise by polynomials. The shape of B-Splines is governed by a set of control points and a knot vector over the range where the curve is defined. B-Splines are highly popular because of simplicity of their construction and their capacity to approximate complex shapes. An additional benefit comes from their property of local controllability [1], i.e. changing one control points doesn't alter whole of the curve but a small region, allowing piecewise fitting of our pattern.

A degree *n* B-Spline curve [17] is defined as

$$S(t) = \sum_{i=0}^{m-n-2} P_i \times b_{i,n}(t) \ , \ t \in [t_n, t_{m-n-1}] \tag{1}$$

for given *m* real values t_i called knots with

$$t_0 \leq t_1 \leq \cdots t_{m-1} \tag{2}$$

P_i are *m-n-1* control points forming a convex hull and $b_{i,n}(t)$ is the blending (De-boor) function given by [15] *Cox-de-Boor recursion formula* as in (3),(4).

$$b_{j,0}(t) = \begin{cases} 1 & if \quad t_j \leq t < t_{j+1} \\ 0 & \quad otherwise \end{cases} \tag{3}$$

$$b_{j,n}(t) = \frac{t-t_j}{t_{j+n}-t_j} b_{j,n-1}(t) + \frac{t_{j+n+1}-t}{t_{j+n+1}-t_{j+1}} b_{j+1,n-1}(t) \tag{4}$$

A. Recognising the Dominant Points

The control points in B-Splines guide the shape of the curve and changing parameter *t* traces the curve. These control points significantly define the positions of some dominant points (points of high curvature) on the curve.

We try to find these dominant points [2], as follows:

- For each point in the data set, the radius of curvature at each point is calculated [20].
- The radius of the unique circle passing through P_i, P_{i-k}, P_{i+k} is considered as radius of curvature at point P_i, where *k* is a small integer and may vary over a range. The average of the radius for different values of *k* can also be considered.
- Curvature is inversely proportional to the radius.
- If the radius at a point is less than a threshold value, the points is taken to be in dominant region.

The point of minimum radius in the neighbourhood of each dominant region is the dominant point (Fig. 3.(b)).

B. Non Uniform Rational B-Splines (NURBS)

NURBS are the B-Splines in which knots are not equidistant. We calculate the knot vector from position of dominant points. The positioning of knot vectors can be treated as defining time spent in drawing each section of curve [3] while hand-drawing.

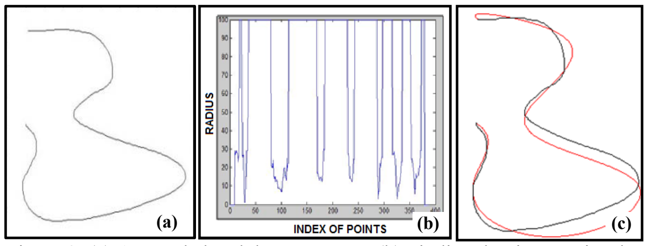


Figure 3. (a) A sample hand drawn pattern; (b) Finding dominant points by looking for local minima in radii; (c) Approximation (red) for the pattern (black) drawn by user.

After a detailed study of the behaviour of knot vectors, it is noticed that the best knot values would be the one which are proportional to occurrence of the dominant points in space [3]. So we use the distance between consecutive control points to find knot values in proportion. With the help of these knot values the corresponding values of De-boor function $b_{i,n}(t)$ in (1) are calculated.

Assuming that if the given curve is made to pass through all the *n* dominant points, and given curve is smooth, the resulting B-Spline will be a good approximation of the input pattern, (1) is solved to determine the control points. We get *n* linear equations using *n* dominant points [3], solving which we can determine the control points.

C. Uniform B-Splines and Error Minimisation

As indicated in results (Fig. 3.(c)), third degree NURBS has approximated the curve shape to large extent. But there is still lot of error. Also, in non uniform B-Spline, knot vector has to be separately computed and stored. Moreover, it is deduced after testing that the usage of non-uniform knot vector hardly makes any difference to image quality. So we switch to uniform B-Splines subsequently.

For 3rd degree uniform B-Splines, knot vector consists of equally spaced *n+4* knots, with vector normalised between '0' and '1', for *n* number of control points. We actually do not need the information on dominant points, however for an approximation of number of control points required, the scheme described in Section III.A above can be used.

We now define an error function [4] as

$$e = \sum_i |q_i - S(t_i)|^2 \tag{5}$$

Where q_i is the data point recorded as input and $S(t_i)$ is the corresponding point on the approximating curve from the definition of B-Splines (6). The error *e* is summed over *m* number of data points in input data over which the error has to be minimised. Parameter t_i is incremented over the range of knot vectors with fixed increments.

$$S(t_i) = \sum_{k=1}^{n} b_{k,3}(t_i) \times P_k \tag{6}$$

To minimize the error, its first derivative must be zero.

$$\frac{de}{dP_j} = 0 \tag{7}$$

This gives:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} b_k(t_i) \times b_j(t_i) \times P_k = \sum_{i=1}^{m} q_i \times b_j(t_i) \tag{8}$$

Equation (8) can be rewritten as product of Matrices as

$$\begin{bmatrix} b_1(t_1) & \cdots & b_1(t_m) \\ \vdots & \ddots & \vdots \\ b_n(t_1) & \cdots & b_n(t_m) \end{bmatrix} \begin{bmatrix} b_1(t_1) & \cdots & b_n(t_1) \\ \vdots & \ddots & \vdots \\ b_1(t_m) & \cdots & b_n(t_m) \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix}$$

$$= \begin{bmatrix} b_1(t_1) & \cdots & b_1(t_m) \\ \vdots & \ddots & \vdots \\ b_n(t_1) & \cdots & b_n(t_m) \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} \tag{9}$$

$$B \times B^T \times P = B \times Q \tag{10}$$

Equation (10) is solved by pseudo-inverse method to get column vector *P*, set of required control points.

The number of control points can be varied as a trade-off between the compression ratio and error. The number of control points can also be varied iteratively [19] from the initial estimate of dominant points from Section III.A till error *e* in (5) reduces below a desired value. Since the knot vector used is uniform, only the array of control points is needed to regenerate the drawn pattern.

IV. RESULTS

Fig. 5 shows the results from the MATLAB implementation of Uniform B-Spline Approximation Scheme as discussed in Section III. The results outstand over all previous implementations. The control points are determined from iterative check on the distortion in terms of square error *e* in (5) till it decreases below a threshold.

B-Splines successfully model even the complex hand-written patterns including sharp turns and kinks. Since more number of control points is required to model sharp features, the Segmentation method discussed in Section II.A, II.B can also be pre-employed as that will further decrease the number of control points. Comparisons made in Table I and II give an estimate of the amount of compression achieved along with the distortion in pattern. (The relative error in Table II is the average error per stroke). Since compression can always be improved by increasing the number of control points, this scheme works as a trade-off between Rate and Distortion. Fig. 4 shows various characteristics of the technique averaged over different types of test patterns (scripts, multistroke, singlestroke etc.)

V. APPLICATION

We have developed an SMS application on android platform to test real-time computing performance of the algorithm. The interface has been kept simple to make the application user friendly and demonstrate the functionality of algorithm. Android uses Linux Kernel version 2.6 for system services such as memory and process management [21]. Each software runs as an individual process having a separate instance of Dalvik Virtual Machine (DVM). Fig. 7 shows the architecture of the application.

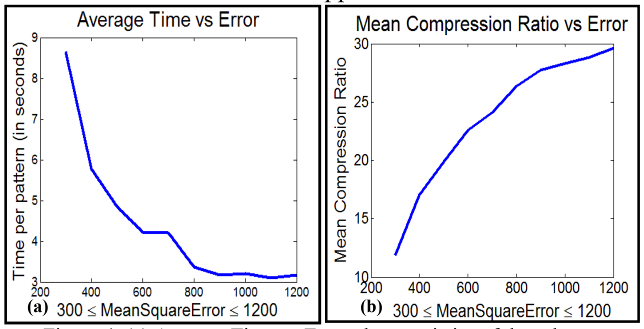


Figure 4. (a) Average Time vs Error characteristics of the scheme. (b) Mean Compression Ratio vs Error characteristics.

Mean Compression Ratio: Average of Compression Ratios taken over a set of test patterns.

In our application, a user can compose a message with stylus on a touch-screen interface. Strokes are recorded and processed (compressed and encoded) as an Asynchronous Task. Drawn figure (Fig. 6) is sent via SMS using Android Telephony Manager. The Receiving end saves the message in a SQLite Database and displays it as a list, clicking on which displays the strokes. For all compression purposes, methods have been written with standard Java Libraries. SMS Send and Receive, Database Management, Views and Thread manipulation have been performed using libraries provided by Android Platform.

Calculating the matrix inverse in (10), is the most time consuming step and has been done using Gaussian Elimination which is of the order $O(n^3)$, where n is the number of control points. Multiplication of matrices for the same equation takes $O(n^2m)$ time where m is the input number of data points in a stroke. Rest of the steps consumes linear time. So overall complexity of the system would be $O(n^3+n^2m)$. Table III shows time performance of the system. For each stroke, error and Mean Opinion Score (MOS) has been calculated, taking opinion of 10 users.

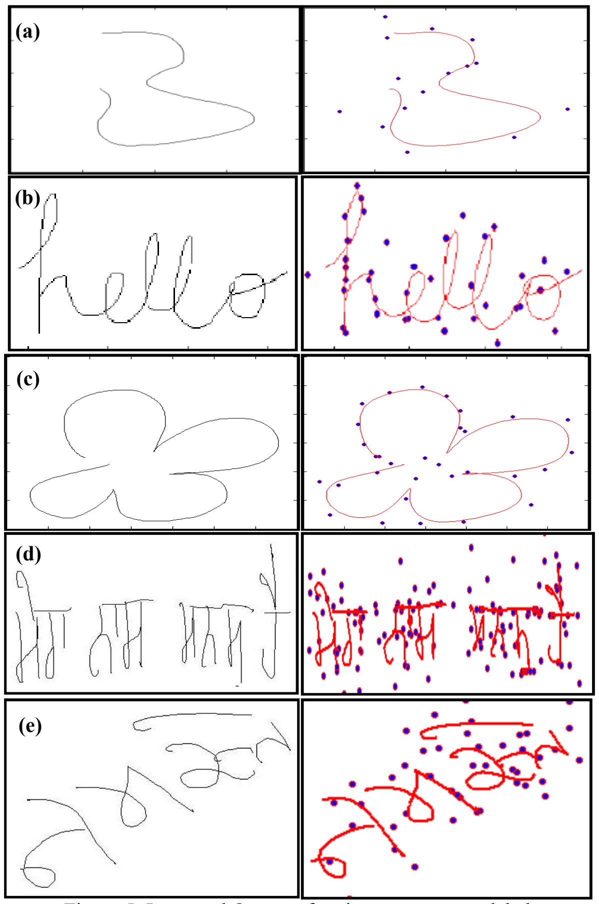


Figure 5. Input and Output of various patterns modeled

Table I: Rate Distortion Error Analysis for Fig. 5(b)

Control Pts.	21	31	33	34	36	41
Error	1566	537	505	454	485	360

VI. Conclusions

We have presented a novel compression scheme to store and transmit data using low bandwidth. The algorithm proposed is independent of user language and can be implemented online for instant messaging applications. B-Spline curves used to approximate user strokes are scalable to any display size [18]. The number of control points is determined by calculating the dominant points in the curve and iteratively reducing error. To demonstrate the feasibility of our system, we have developed an Android based SMS Application that works well in real time scenario.

Performance of the scheme worsens when the hand-written pattern consists of a number of random scratches, small dots or strokes, as in this case a large number of points have to be stored to represent a little data. However, this problem can be tackled by incorporating appropriate heuristics in our scheme.

Table II: Comparison of the results obtained with standard schemes.

Fig. 5	No. of Data Points	No. of Control Pts	Relative Error	I/P Strokes (kB)	O/P Control Pts (Bytes)	Huffman Coded (Bytes)
(a)	387	15	480	3.34	118	49
(b)	230	39	403	1.92	332	146
(c)	438	31	487	3.85	250	104
(d)	620	154	342	5.68	1177	490
(e)	195	48	456	1.5	414	172

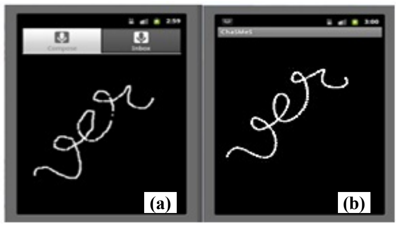


Figure 6. (a) Input; (b) Output, interface on Android Emulator

Table III: Time analysis on Android Emulator

S. No.	Input pts. (m)	Control pts. (n)	Time (s)	Error	MOS
1	230	10	1.276	423	4.2
2	384	15	2.293	521	4.3
3	584	18	3.367	367	4.5
4	716	20	4.612	652	3.4

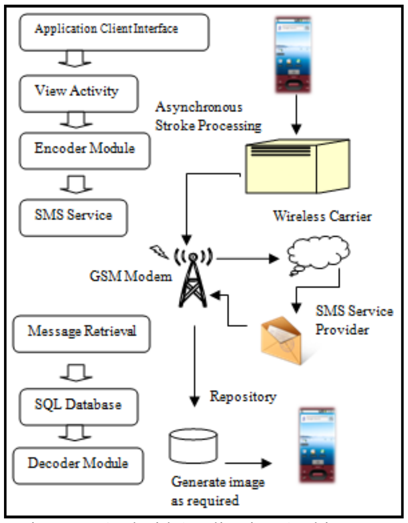


Figure 7. Android Application Architecture

References

[1] F. S. Cohen, Z. Huang, and Z. Yang, "Invariant Matching a.nd Identification of Curves Using B-Splines Curve Representation", IEEE Transactions on Image Processing, Vol. 4, No. 1, Jan 1995

[2] W Wu, "Dominant point Detection using adaptive bending value", Image and Vision Computing 21 (2003), vol 21, no. 6, pp. 517-525

[3] Z. XiuYang, Y. YanSheng and Y. Bo, "Dominant point detecting based non-uniform B-spline approximation for grain contour", Sci China Ser E-Tech Sci, Feb. 2007, vol. 50, no. 1, pp. 90-96

[4] W. Wang, H. Pottmann, Y. Liu, "Fitting B-Spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization", ACM Transactions on Graphics, Vol. 25, No. 2, Apr. 2006, pp. 214–238.

[5] S. Chaudhury and A. Khandelia, "Handwriting Based Interface for Communication", Ist International Workshop on Expressive Interactions for Sustainability and Empowerment (EISE 2009), 29 - 30 October 2009, London, UK

[6] Z. Liu, H. S. Malvar, and Z. Zhang, "System and method for ink or handwriting compression", US Patent No 7,302,106 B2, Nov 2007.

[7] S. Masaki, M. Kobayashi, O. Miyamoto, Y. Nakagawa, T. Matsumoto, "An on-line handwriting character recognition algorithm RAV (reparameterized angle variations)", 4th ICDAR, Proc. vol.2, pp. 919–925, 1997

[8] M. Hamdani, H. El Abed, M. Kherallah, and A. M. Alimi, "Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition" 2009 IEEE DOI 10.1109/ICDAR.2009.40, pp. 201 - 205

[9] Zhengyou Wang, "High Efficient and Real Time Huffman Codec Used in Handwriting Short Message Service", 3rd International Conf. on Natural Computation 10.1109/ICNC.2007.416, pp. 132 - 136

[10] T. Wakahara, "On-Line Handwritten Character Recognition Using Local Affine Transformation," Systems and Computers in Japan, vol. 20, no. 7, 1989

[11] C.C. Tappert, C.Y. Suen, and T. Wakahara, "The State of the Art in Online Handwriting Recognition", IEEE Transaction Pattern Analysis and Machine Intelligence, vol. 12, no. 8, pp. 179-190, August 1990

[12] R. Plamondon, "Online and off-line handwriting recognition: a comprehensive survey," Transactions on Pattern Analysys and Machine Intelligence, vol. 22, Jan 2000

[13] C. D. Worth, "xstroke: Full-screen Gesture Recognition for X," in Annual Technical Conference, FREENIX Track, pp. 187–196, USENIX, Apr 2003.

[14] B.K. Sin and J.H. Kim, "Ligature modeling for on-line cursive script recognition", IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) (6), pp. 623–633

[15] Carl de Boor (1978). A Practical Guide to Splines. Springer-Verlag. pp. 113–115

[16] Guerfali W, Plamondon R, (1998) "A new method for the analysis of simple and complex planar rapid movements". J Neurosci Meth 82(1): pp. 35–45

[17] Wikipedia.org, "B-Spline", from "http://en.wikipedia.org/wiki/B-spline cite_note-3" , Retrieved on 3 Feb, 2010

[18] A. Quint, Scalable Vector Graphics. IEEE Multimedia - Vol. 3, pp.99-101, 2003

[19] J. Yang and H. Byun, "Curve Fitting Algorithm Using Iterative Error Minimization for Sketch Beautification". ICPR 19th International Conference, 2008.

[20] Y.Qiao and G. Leedham, "Segmentation and Recognition of Handwritten Pitman Shorthand Outlines using an interactive heuristic Search", Pattern Recognition, vol 26, pp. 433-441, 1993

[21] "Android Developers", from "http://developer.android.com", Retreived on 5 Decemeber, 2010 and 20 January 2011.