SIGNATURE AUTHENTICATION

By

Romit Beed, Debapriya Ghosh, Farhana Javed Zareen, Nikita Goyal

Post Graduate Department of Computer Science,

St. Xavier’s College(Autonomous), Kolkata

ABSTRACT

Signature can be seen as an individual characteristic of a person which, if modeled with precision can be used for his/her validation. An automated signature authentication technique saves valuable time and money. The paper is primarily focused on skilled forgery detection. It emphasizes on the extraction of the critical regions which are more prone to mistakes and matches them following a modular graph matching approach. The technique is robust and takes care of the inevitable intra-personal variations. The results show significant improvement over other approaches for detecting skilled forgery.

1.INTRODUCTION

A handwritten signature as a behavioural biometric is the mean accepted method to declare someone's identity. Many documents necessitate a handwritten signature. In general, there are two ways to process the signature sample. The first is on-line, where the image is captured directly as handwriting trajectory. The second is off-line, in which we use a digitizer in order to acquire a digital image.

Signature Verification[1][2][3][7] is the process of recognizing an individual’s handwritten signatures. Signatures have been by far the most popular means for establishing the authenticity of individuals. Signature authentication[1][2][3][7] offers a quick, simple and cost effective means for validating the authenticity of a document by determining the difference between an original signature and a counterfeit one.

This project takes on the task of comparing signature which are store as image files in a folder. The purpose is to provide easy and fast matching procedure with highly accurate search results. This paper covers on offline verification of signature. It authenticates signature based on certain traits such as critical region extraction and matching.

The front end used is Matlab® (Mathworks IncLtd). MATLAB solutions are expressed in familiar mathematical notation. Typical uses includes: Math and computation, Algorithm development, Data acquisition, Modelling, Simulation, and prototyping, Data analysis, exploration, and visualization, Scientific and engineering graphics, Application development, including graphical user interface building,MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows formulating solutions to many technical computing problems, especially those involving matrix representations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or FORTRAN.

2.ALGORITHM DESIGN

The input image goes through the following procedure before judging that the two signatures are accurate or not. The steps are Binarization ,Noise Removal, Rotation, Thinning, Critical point extraction, Critical region matching and Verification. Each step is mentioned in a separate section starting from section 2.1 to to 2.7. Figure 1 shows the approach in form of a diagrammatic representation.

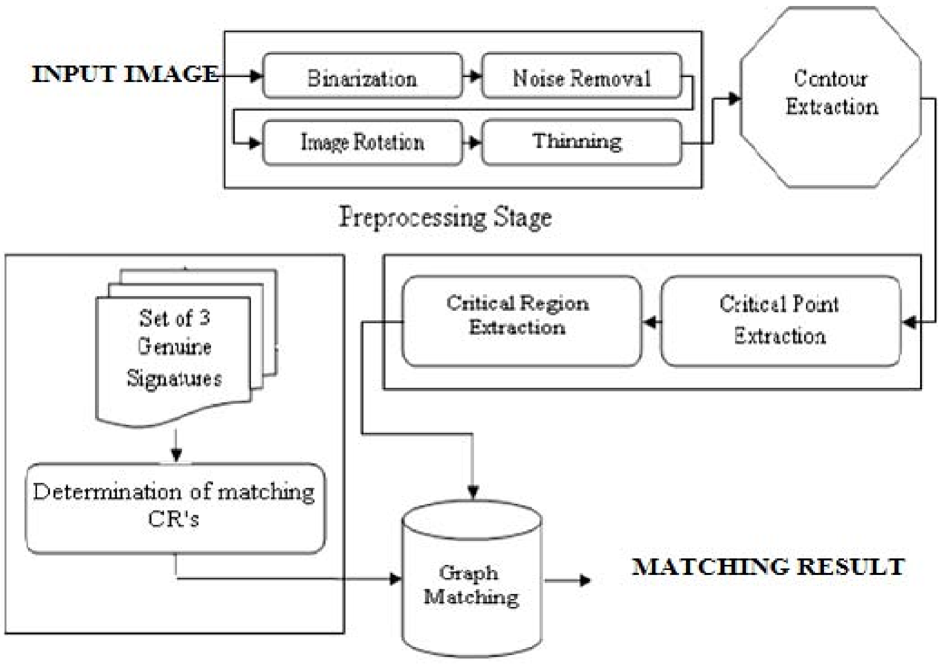


FIG1: BLOCK DIAGRAM OF THE PROPOSED APPROACH

2.1 BINARIZATION

The main components of binarization include statistical analysis of images, determination of a threshold value based upon the statistical results and applying the threshold value to gray-lcvel images. Statistical analysis of gray- level images may include determination of mean, variance. Standard deviation, contrast stretch, histogram etc. or it can be a combination of any of these Determination of a threshold value is very much important and perhaps the most sensitive part of any image binarization scheme because a wrong value of threshold may result in losing some image information (an object can be considered as part of background and vice versa). Moreover, the value of threshold should be sensitive with the overall contrast stretch of the corresponding image. The threshold value is applied to the image to represent it in I-bit after the proper determination of a threshold limit.

The simplest method to convert a gray scale image into a binary image is to compute mean of the image and set the value of mean as the threshold for binarization. But this approach has many shortcomings, as it may not take care about the features and objects in the image properly. This is due to the fact that the mean of an image may be disturbed drastically by the addition of noise pixels or very few numbers of pixels having the intensity close to any of the boundaries of gray scale.

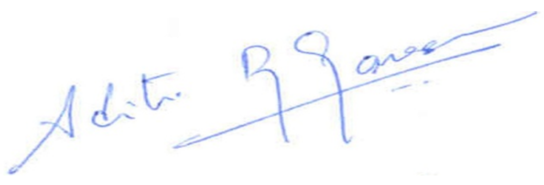


FIG 2.1a ORIGINAL IMAGE

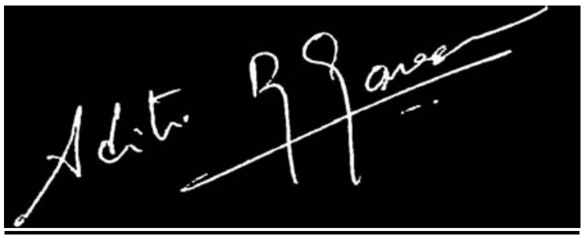


FIG 2.1b BINARIZED IMAGE

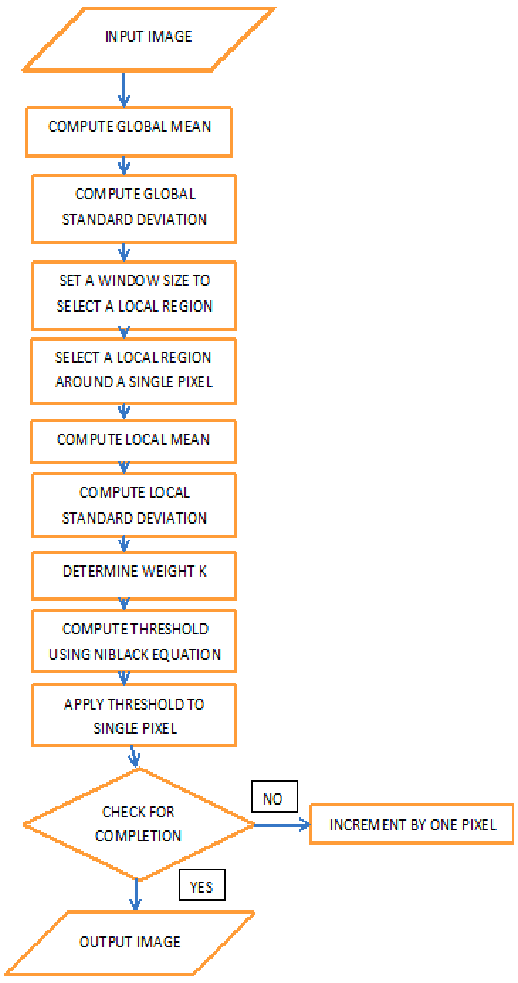

INPUT IMAGE
COMPUTE GLOBAL MEAN
COMPUTE GLOBAL STANDARD DEVIATION
SET A WINDOW SIZE TO SELECT A LOCAL REGION
SELECT A LOCAL REGION AROUND A SINGLE PIXEL
COMPUTE LOCAL MEAN
COMPUTE LOCAL STANDARD DEVIATION
DETERMINE WEIGHT K
COMPUTE THRESHOLD USING NIBLACK EQUATION
APPLY THRESHOLD TO SINGLE PIXEL
CHECK FOR COMPLETION
NO
INCREMENT BY ONE PIXEL
YES
OUTPUT IMAGE

FIG 2.1a FLOW CHART FOR BINARIZATION

2.2 NOISE REMOVAL

Once we have binarized an image, the noise components must be removed. Small components (pixel_size<5-10pixels) are removed by using a simple morphological filter. Assumption that the signature content would be more prevalent in the image, the image is passed though a low pass filter to eliminate the low frequency noise components. The filtering is done by using 2-D convolution with a 5X5 Unity matrix. The results are illustrated in figure 2a and 2b. This process converts the binary image into a gray scale image. The image thus obtained is further binarized sing a strict estimated threshold. We use Niblack's Algorithm [4] for this.

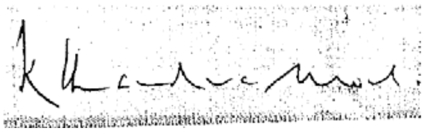


FIG 2.2b NOISY IMAGE

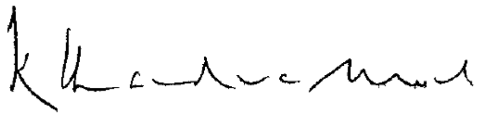


FIG 2.2c NOISE REMOVED IMAGE

2.3 ROTATION

The accuracy of the results is largely dependent on the rotation algorithm used for orientation correction. The rotation algorithm should be robust and must produce the same results for images taken from the same user. The rotation algorithm *rotate-image* is given below. The binarized and noise cleaned signature image is input to the algorithm. We use the bottom pixels of a signature image as a template to fit an *orientation line* through them using the *polyfit* function of Matlab® (Mathworks Inc Ltd). The *polyfit F*unction is further explained in Section 3.1. Finally, the *cp2tranform ()* function produces a projective transformation of the input image, using the slope of the orientation line as a guiding parameter. Experimentally, we found that the above algorithm showed excellent parity between the rotated-corrected transformations of the sample signature and new input signature from the same user, when the rotation angle varied between -30 to +30 degrees. The Rotation algorithm is in section 3.

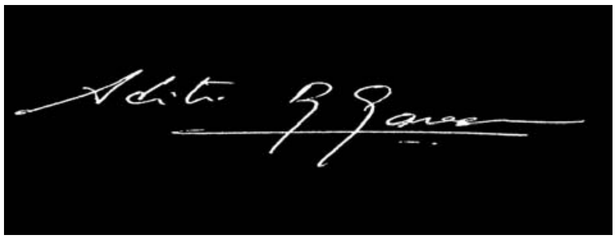


FIG 2.3a IMAGE BEFORE ROTATION

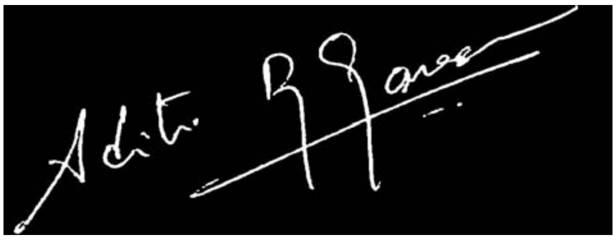


FIG 2.3b IMAGE AFTER ROTATION

2.4 THINNING

Hilditch thinning algorithm[5] is widely used as a useful method of pre-processing in image process. There are two versions for Hilditch's algorithm[5], one using a 4x4 window and the other one using a

3x3 window. Here, the 3x3 window version is considered and the algorithm is described below. For a pixel p0, the aim of thinning algorithm is to decide whether to keep it as part of the result skeleton or delete it from the image. For this purpose, the 8 neighbors of a pixel p0 must be investigated, when pixel p0 is part of a skeleton, pixel p0 will be deleted or not deleted according to certain conditions. But during one path process, the value of pixel p0 should be set to another value such as −1 according to the Hilditch's algorithm[5], it will be set to 0 until all pixels in the image have been investigated during this path. Then this process is repeated until no changes are made.

2.5 CRITICAL POINT EXTRACTION

The proposed approach consists of extracting critical points on the input signature, locating the corresponding critical points among the sample signatures, extracting the critical regions centered around the critical points on the respective signatures, matching the corresponding critical regions using graph matching algorithm, training the sample signatures and finally, verifying the authenticity of the test signature.

Polyfit Function: The digitally scanned 2-D image of a signature gives a pixilated image. To overcome this limitation, Matlab's Image Processing Toolbox is used to estimate the continuous curve that best fits the image. The equation of any image can be estimated using the *polyfit* function.

A contour based approach is followed to extract the critical points. In this approach the contour is traversed and any sharp change in the curve is marked as a critical point.Critical points can be best described as the set of points which model the basic structure of the signature. They are a minimum set of points to represent the shape of a signature. A contour can be described as the outer boundary of a signature. To extract the same,the disconnected components in the signature are joined and the 'holes' inside the signature are filled. The set of all four-connected boundary pixels define a contour of the signature. The process undergoes the following steps. The contour image obtained is first thickened using a 5X5 morphological filter followed by thinning. This is done to eliminate any sharp changes and to bring about uniformity in the curve.

A unique point on the contour image (must occur in the same region for the same user set) is then selected as a starting point and the contour is traversed in the clockwise direction. Critical points are encountered during this traversal by an algorithm Critical points extract. A brief explanation is as follows. The algorithm repeatedly segments the signature image small curves using the *polyfit* function, taking care that at least 5 points are used. As the curve is extended to include newer points, the deviation of the curve as given by the error value *abs(S.normr)* indicates whether a peak is encountered. A critical point is identified when either the current peak exceeds an experimentally tuned threshold, or when the number of peak points obtained past the last critical point exceeds a pre-determined number, as evaluated by the *track_error_peak* function. The algorithm is given in section 3.

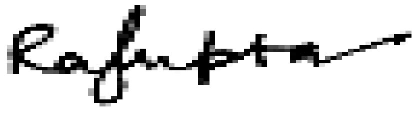


FIG 2.5a INPUT IMAGE

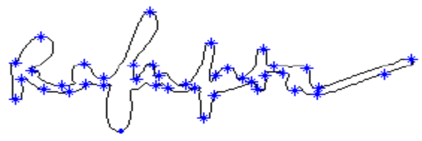


FIG 2.5b CRITICAL POINT EXTRACTED

2.6 CRITICAL REGION MATCHING

After extracting the critical points from the sample signatures, the next step requires finding out the correspondence among the critical points in these signatures. The aim is to find out which critical point in a signature A corresponds to which critical point in the signature B. The procedure is explained below.First, each critical point on signature *SigA* and signature *SigB* is masked with a 21 ×21 *black block* centered on the critical point. Every pixel within a block is set to the value '1' (black). The remaining pixels in both images are marked '0' (white). Thus we obtain two new *images SigA'* and *SigB'* respectively containing only the black boxes at their respective positions. Let *NSigA* and *NSigB* be the total number of critical points on SigA and *SigB* respectively.The algorithm next finds the common portion of the every block of *SigA'* with respect to each of the blocks extracted from the *SigB'* by using a simple AND gate function between corresponding locations, operating on the binary values 0 (white) and1(black).Finally, for each block in *SigA'* the maximally overlapping block in *SigB'* is located.In effect, the algorithm traverses an *Overlap_matrix* with *NSigA* rows and *NSigB* columns,where cell(i,j) is set equal to the number of overlapping pixels between the *ith* block of *SigA'* and *jth* block of *SigB'*. The highest value cell in row *k* indicates the matching critical point of *SigB* for the *kth* critical point on *SigA.*

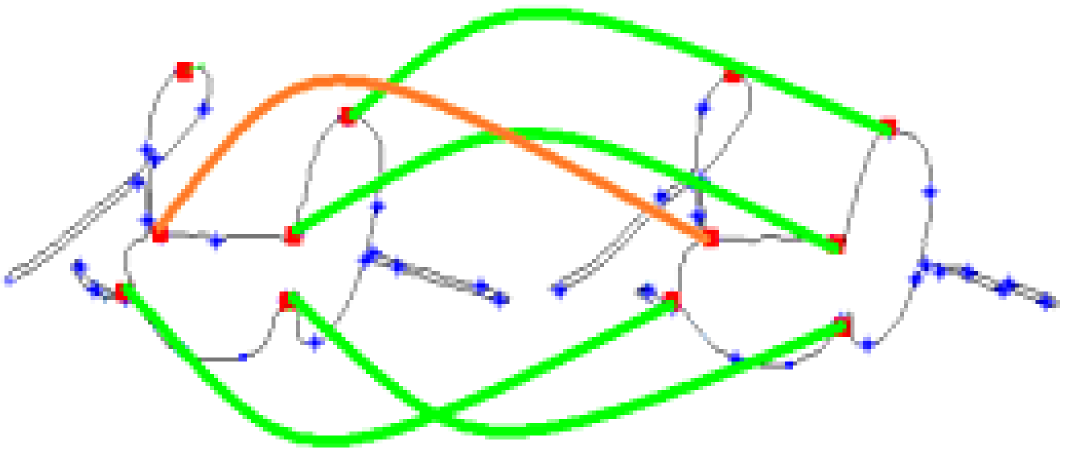


FIG 2.6 MATCHING

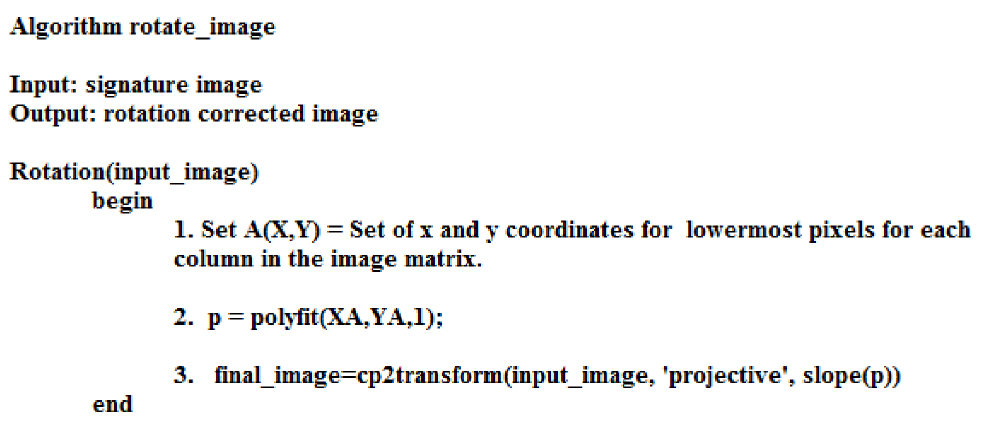
2.7 VERIFICATION

After determining the one-to-one corresponding matched critical points in the sample signatures, their respective critical regions are extracted. Critical regions serve as a sound basis for modular graph matching. Instead of using the entire signature image, its critical portions are extracted and corresponding regions are compared to judge the overall similarity between the input and sample signatures.The algorithm *Ext&Mat_Critical_Regions* for extracting and matching corresponding critical regions in a pair of sample signatures. Its working is outlined below.A critical region is a 31× 31 block extracted from a signature image and containing a critical point at the center. For every critical point on signature A, the algorithm extracts a 31×31 block *CR1*, taking the critical point *cp1* as centre. The corresponding 31× 31 block *CR2* from signature B is also extracted, taking the matching critical point *cp2* as centre. Each critical region is represented by a undirected graph in which every black pixel in the critical region signifies a vertex. The *x, y* coordinates of all black pixels in *CR1* and *CR2* represent vertex sets *S1* and *S2* respectively. Matching two graphs measures the similarity of the two corresponding critical regions based on their geometrical shapes. The distance-matrix *W* is a (m × n) adjacency matrix whose rows represent vertices of *S1* and whose columns represent vertices of *S2* (where |*S1]*>=|*S2*|). We calculate the Euclidean distance each pair of vertices in *S1* and *S2* using the x-y co-ordinates of their corresponding pixels.

The formulated assignment problem is solved using the Hungarian method[6] . This returns the optimal cost/distance *min_dist* between critical regions *CR1* and *CR2*. The *min_dist* is divided by |*S1*| to get a normalized minimum distance per pixel. It is further divided by a factor α which is a measure of the percentage of vertices matching in *S1*and *S2*.The above procedure is repeated for every pair of matching critical regions to yield the array of optimal distances *Optimal_Distance*. Once we get the optimal distance vector we compare it against a threshold (opt_thresh~=15). For each value less than the threshold the vote number is incremented. For a set of N values any signature giving more than two third votes is considered a genuine signature. In case a signature gives consideration results with the first genuine sample but is not good enough to be tagged as genuine, in that case the signature is tested against the second genuine signature sample. If it still does not pass the test, then is tested against the third sample. If the signature gives average results (2N/3>votes>N/3) with the entire genuine signature-set, then it is tagged as a probable forgery. If at any stage the input signature gives unacceptable results (votes<N/3) with any of the genuine signature samples, then it is straightaway rejected.

3.ALGORITHMS

3.1 ROTATION



```
Algorithm rotate_image

Input: signature image
Output: rotation corrected image

Rotation(input_image)
    begin
        1. Set A(X,Y) = Set of x and y coordinates for  lowermost pixels for each
        column in the image matrix.

        2. p = polyfit(XA,YA,1);

        3. final_image=cp2transform(input_image, 'projective', slope(p))
    end
```

3.2 CRITICAL POINT EXTRACTION

```
Algortihm Critical Points Extract
Begin
    Initialize pixel_count to 0
    Initialize cp_count to 0
    Initialize error_vector to NULL
    Initialize norm to 0
    Initialize V1 to v1
    Initialize V2 to v2
    for i = 0 to size of contour_array,loop
        Increment pixel_count by 1
        If pixel_count exceeds V1 then
            Curve = generate curve using polyfit(x_coord(initial to initial+pixel_count),
                y_coord(initial to initial+pixel_count))
            norm = deviance of curve
            error_vector = error_vector U norm
        End If
        If abs(norm) < V2
            check if current point has exceeded an experimentally tuned threshold value.
            If YES,then
                critical_point_x(cp_count)=x_coord(initial+pixel_count)
                critical_point_y(cp_count)=y_coord(initial+pixel_count)
                Increment cp_count by 1
                initial=initial+pixel_count
                Reinitialize pixel_count to 0
                CONTINUE
            End If
        Else
            critical_point_x(cp_count)=x_coord(initial+pixel_count)
            critical_point_y(cp_count)=y_coord(initial+pixel_count)
            Increment cp_count by 1
            initial=initial+pixel_count
            Reinitialize pixel_count to 0
        End If
    Next
    Return critical_point_x() and critical_point_y()
End
```

3.3 CRITICAL REGION MATCHING

```
Algorithm Critical Points Match
Input : cp_sig1[ ],cp_sig2[ ]
Output : M[ ]-a set of matched critical point coordinates
Begin
    Initialize sig1_block[800][800] to 0
    Initialize sig2_block[800][800] to 0
    Initialize overlap[s1][s2] to 0
    Initialize max[s1] to 0
    Initialize stucture M(x_s1,y_s1,x_s2,y_s2)
    for i = 0 to size of cp_sig1[], loop
        All points in sig1_block[x coordinate of cp_sig1[i] to x coordinate of cp_sig1[i + 20]][y coordinate of
cp_sig1[i] to y coordinate of cp_sig1[i + 20]] are made 1
        for j= 0 to size of cp_sig2[], loop
            All points in sig2_block[x coordinate of cp_sig2[j] to x coordinate of cp_sig2[j + 20]][y
coordinate of cp_sig2[j] to y coordinate of cp_sig2[j + 20]] are made 1
            overlap_value=sig1_block[ ][ ] AND sig2_block[ ][ ]
            overlap[i][j]=overlap_value
            Reinitialize sig2_block[ ][ ]
        Next
        Reinitialize sig1_block[ ][ ]
    Next
    Initialize Count to 0

    for i = 0 to size of cp_sig1[ ],loop
        max_val = maximum value in overlap[i][0 to j]
        if max_val exceeds a predetermined threshold value for matching,then
            j1 = index,where max_val was found
            M(count) = (x of cp_sig1[i],y of cp_sig1[i],x of cp_sig2[j1],y of cp_sig2[j1])
            Increment count by 1
        End If
    Next
    Return structure M
End
```

3.4 VERIFICATION

```
Algorithm Critical Region Extract and Match
Input:-Contour Images Sig1[ ][ ],sig2[ ][ ],structure Match(x1,y1,x2,y2)
Output:-Optimal Distance Matrix[|M|]
Begin
    Initialize cr1[31][31],cr2[31][31] to 0
    Initialize set1[ ],se2[ ] to 0
    Initialize dist_mat[ ][ ] to 0
    Initialize optimal_dist[|M|] to 0

    for i=0 to size of structure Match(x1,y1,x2,y2),loop
        X1,Y1 and X2,Y2 = ith entry of Match()
        cr1[ ][ ]=values in sig1[X1 - 15,X1,X1+15][Y1-15,Y1,Y1+15]
        set1[ ]=coordinates in cr1[ ][ ] which are black
        cr2[ ][ ]=values in sig2[X2 - 15,X2,X2+15][Y2-15,Y2,Y2+15]
        set2[ ]=coordinates in cr2[ ][ ] which are black
        Reinitialize dist_mat[ |set1| ] [ |set2| ] = 0
        for d = 0 to |set1| , loop
            for g = 0 to |set2| , loop
                dist = eucleidian distance between set1[d] and set2[g]
                dist_mat[d][g] = dist
            Next
        Next
min_dist = Hungarian_algorithm(dist_mat[ ][ ])
        optimal_dist[i] = min_dist
    Next
    Return optimal_dist [ ]
End
```

4. CONCLUSION

We have proposed an algorithm that not only works better than the similar graph-based offline verification approaches but also works on a sample base of just three authentic signatures, which is closer to the real world requirements. In this paper we demonstrate that it is possible to achieve very low error rates even for skilled forgeries. The approach is computationally faster as compared to other graph matching techniques. It introduces the concept of modular graph matching.

5. REFERENCES

[1] N. Christofides, *Graph theory: an algorithmic approach* (New York, Academic Press Inc., 1977).

[2] Ibrahim S.I. Abuhaiba, Offline Signature Verification Using Graph Matching, *Turk J ElecEngin, VOL.15, NO.1* 2007.

[3] Siyuan Chen and Sargur Srihari, A New Offline Signature Verification method based onGraph Matching , *18th International Conference on Pattern Recognition Volume 2,* Issue ,2006.

[4]. Lal Chandra, Puja Lal, Raju Gupta, Arun Tayal,Dinesh Ganotra: Improved adaptive binarization technique for document image analysis. VISAPP (1) 2007: 317-321.

[5].http://en.wikipedia.org/wiki/Hilditch_algorithm

[6]. http://en.wikipedia.org/wiki/Hungarian_algorithm

[7].http:// Wikipedia.org/signature verification.