New Segmentation Algorithm for Offline Handwritten Connected Character Segmentation

U.K.S. Jayarathna[1], G.E.M.D.C. Bandara[2]
1. Department of Statistics and Computer Science, Faculty of Science, University of Peradeniya, Peradeniya, Sri Lannka.
2. Department of Production Engineering, Faculty of Engineering, University of Peradeniya, Peradeniya, Sri Lannka.
Email:sampathid2003@yahoo.com, dcb@pdn.ac.lk

***Abstract* - An approach for the Segmentation of offline handwritten connected two-digit strings is presented in this paper. Very often even in a printed text, adjacent characters tend to touch or connect. This makes it a problem in performing proper character isolation, hence difficult in segmenting the digit strings in order to recognize its individual characters. We, in our study have developed an algorithm, which provides a solution based on the analysis of the foreground pixel distribution to segment, connected digit string pairs.**

In the segmentation stage, the junction based splitting technique decides complete segments of the connected digit strings. Use of fuzzy characteristic values at the merging of the complete segments isolates the major segments (Merged complete segments) from the minor segments. In this work, it was found that the unwanted connection resides within the minor segments of the connected character skeleton. At the character isolation stage, all major segments are combined with each of minor segments to generate set of different connection sketches. In order to recognize individual characters of the connection string, these sketches are formulated into a new input character image, which then can be used as an input for a character recognition system.

Keywords: Skeletonization, Fuzzy Rules, Connected digit Strings, Binarization, Segmentation

I INTRODUCTION

Alphabetic character segmentation and recognition is a basic need in incorporating brainpower to computers. Machine intelligence involves several aspects among which optical recognition is a tool, which can be integrated to text recognition and speech recognition.

Offline handwritten character reading by computer program is a complex undertaking. It is essential to separate a given character string correctly into the sequence of characters. Any failure or error in the segmentation step directly produces a negative effect on recognition. The difficulty of handwritten character segmentation comes from the great variety of handwritings. In the case of hand-printed scripts, segmentation is a relatively simple task. In the case of overlapped scripts, broken characters, connected characters, loosely configured characters, and mixed scripts, segmentation is difficult. Overlapped, broken, connected and loosely configured characters are major causes of segmentation errors [2]. Very often even in printed text, adjacent characters tend to touch or connect.

The Segmentation and Recognition of Connected characters, is a key problem in the development of OCR/ ICR systems. Many methods have been proposed in the recent years. These methods include, segmentation of the connected handwritten two digit strings based on the thinning of background regions [3], comparison with touching character images synthesized from two single character images [4], a neural network based approach to deal with various types of touching observed frequently in numeral strings [5], statistical and structural analysis combined with peak-and-valley functions to segment machine-printed addresses [6]. Performances of the statistical methods depend on the amount of data used in the definition of the statistical model parameters, and are not flexible enough to adapt new handwriting constraints. Once the networks are trained, the advantages of neural networks are automatic learning and quick classification. Their drawbacks are the requirement of long processing time and a proper dataset for learning. The background thinning based approach first locates several feature points on the background skeleton of a connected digit image. Possible segmentation paths are then constructed by matching these feature points. Fuzzified decision rules, which are generated from training samples, are used to rank segmentation paths. The advantage of the work of [3] is the possibility of segmenting single and double touched connection in the character recognition dilemma. But when considering the connected character isolation, in order to recognize a connected character input image, is rather difficult to achieve with the background skeleton analysis. The need of external comparison with the touching characters [4] is difficult to extend in to various different handwritten styles.

II METHODOLOGY

When considering existing systems, there are some major features defined in [3] which, a connection of adjacent handwritten digits can be categorized into (Fig. 1,)

(a) Two strokes from two adjacent digits touching end to end.
(b) The end of a stroke of the left/right digit touching the side of a stroke of the right/left digit.
(c) Overlapping of two vertically oriented strokes from two adjacent digits.
(d) Stroked connection between adjacent two-digit strings.

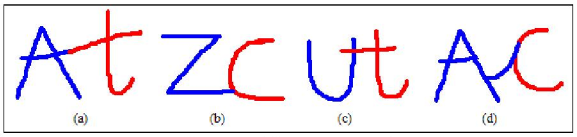

Fig. 1. Different connection styles

The present technique can be used for the detection and the segmentation of type (b) and (c) style connections, while this paper focuses on the performance of the algorithm in unwanted connection segments (d) in the connected character segmentation dilemma. In addition, while it is assumed in this paper that each connecting character image consist of only two component characters, the present technique can be easily extended to deal with the connected character image which consist of three or more characters. The block diagram of the proposed system is depicted in Fig. 2,

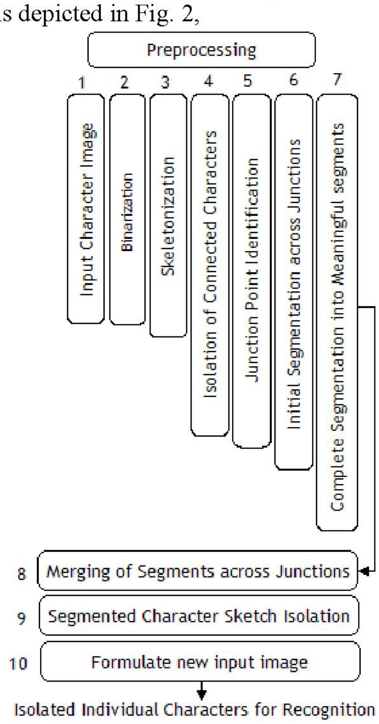

Fig. 2. Block diagram of the proposed system

This paper will cover the proposed system by presenting the preprocessing of the connected character image, namely, the binarization & skeletonizaiton, isolation of connected characters, Junction based segmentation, in sections A, B and C respectively. Merging of the segments across junctions, Segmented Character Sketch isolation and the Formulation of the new Input character image (isolated individual characters) are described in section H. In this work, the same methodology proposed in [8] for isolated offline handwritten character recognition was used to identify each of the individual character in the new input character image. Section IV outlines some experimental results. The paper concludes with conclusions and future work outlines.

A Binarization & Skeletonization

Any scanned digital image is represented as a collection of pixels having intensities within the range of 0% to 100%. Prior to processing the image for skeletonization, the image should undergo a binarization process. In this work, binarization process proposed by the work in [8] is used for the preprocessing of the connected character image. In the binarization process, if the intensity of a particular pixel is less than a particular threshold value, it is set to black (0) and if the intensity value is greater than or equal to the threshold, it is set to white (255). After binarization, every pixel in the image was represented as either black or white. In this work, the threshold value was taken as 200.
The properly binarized image was taken in to account for the skeletonization process, which dealt with getting the single order pixel skeletons of the connected character pattern within the input image. With this work, the foreground pixel based implementation [8] of the "Improved parallel thinning algorithm" is used to obtain the connected character skeleton of the input image.

B Isolation of Connected Character Skeleton into Correlation Area

The skeletonized character image was then processed for individual and connected character isolation. In this work, a simple method is proposed to isolate the connected characters skeletons into c*orrelation area.* For the individual characters; the method proposed in [1] is used.

Definition 1: A *correlation area* is a rectangular area in the image, which contains connected character skeleton. Fig. 3, depicts how a correlation area can be represented.

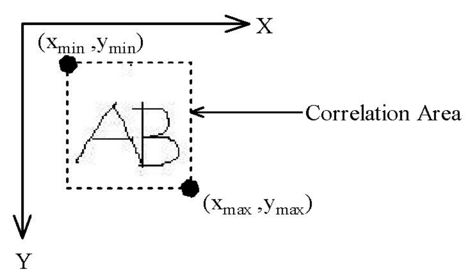

Fig. 3. The representation of the correlation area of a connected two-digit string

In the proposed method, firstly, the rows of the character skeletons are used to isolate, assuming that the maximum distance between two rows of connected character skeleton is 50 pixels. Then the character skeletons in each row are proposed to isolate assuming that the maximum distance between two separate connected character skeletons in a single row is 200 pixels. This process isolates the connected character skeletons into their *correlation area.*

C Junction based segmentation

The most challenging task associated with this work was the segmentation of connected characters skeleton into a set of meaningful segments for the calculation of fuzzy characteristics. Once a skeleton is properly segmented, each of the resulted segments can be used to calculate a set of fuzzy features (MHP, MVP, etc.) as described in [1].

In order to calculate these fuzzy features more accurately, each resulted segment should be a meaningful segment. That is, each resulted segment should be a meaningful straight line or a meaningful arc as described in [8], but not an arbitrary segment.

When considering the connected character segmentation, it is almost impossible to segment connected character skeleton in to a set of meaningful segments. This is due to the possibility of having different connections between two-digit character strings. Therefore the segmentation stage of the proposed work consists of two phases namely, the initial segmentation phase and the total segmentation phase. In the initial segmentation, the input character image undergoes Junction Point identification and Junction Based Segmentation.

Definition 2: A *Junction Point* is a pixel point in the *Correlation area*, having three or more neighboring pixels. It is assumed here that the skeleton is in one pixel thickness.

After the identification of the all junction points (Fig. 4,) in the correlation area, each is used to segment he connected character skeleton in to initial segments.

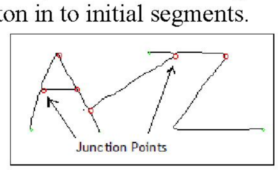

Fig. 4. Junction points of the correlation area

In this research it was found that, the initial segment would not produce complete segmentation due to the non-Junction point connection between segment skeletons. As an example, the handwritten character skeleton 'Z' (in Fig. 5,) would not be segmented into a "negative slanted" and two "Horizontal Lines" because of the non-Junction point connection in between.

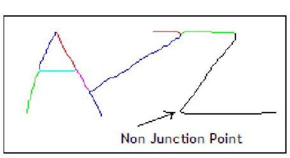

Fig. 5. Initial Segmentation of the correlation area

To compensate for this, a separate segmentation algorithm is used with the rule based segmentation approach proposed in [7] (Fig. 6,).

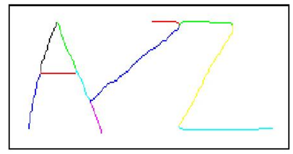

Fig. 6. Complete segmentation in to meaningful segments

D The algorithm

Two types of major data structures are used in this algorithm, namely, the *Point,* which holds the X and Y coordinate values of a pixel point and the *Segmen*t, which is a *Point* array. The input for this connected character segmentation algorithm is the *correlation area* of a connected character skeleton. In order to get the expected segmentation results, the input connected character skeleton should be in one pixel thickness and it should not contain any spurious branches. This section outlines the main routine of the algorithm. Apart from that some definitions proposed in [1] are applied here. These definitions will be used through out this paper to describe the algorithm in detail.

Definition 3: A *starter point* is a pixel point on the character skeleton with which the traversal through the skeleton could be started. *Starter points* are twofold; *major starter points* and *minor starter points*

Definition 4: A *major starter point* is a *starter point,* which is identified before starting the traversal through the skeleton. The identification of *major starter points* is described in Section 5.

Definition 5: A *minor starter point* is a *starter point,* which is identified during the traversal through the skeleton.

The main routing of the connected characters segmentation algorithm (hereafter refereed to as *ccs algorithm*) can be described as follows;

```
Function ccs_algo (correlation_area) returns
total_Segments
    Begin
    major_starters = empty // a queue of Point, major starter points.
    junction_points = empty // a queue of Point, junction points.
    partial_Segment = empty // an array of Segment
    init_Segments = empty // an array of Segment, all partial segs
    complete_Segments = empty // an array of Segments
    total_Segments = empty // an array of Segments
major_starters = find all major starter points (correlation_area)
junction_points = find all junction points (correlation_area)
for each of the initial major start points in major_starters do
partial_Segments = init_Split(major_starter_point,
```

```
junction_points)
add all the segments in the partial_Segments to init_Segments
end for.
        noice_removal(init_Segments)
        complete_Segments = complete_Split(init_Segments)
total_Segments = get_Segment_naming(complete_Segments)
return total_Segments
End
```

E Identification of Starter points and Junction Points

The algorithm starts with finding all the *major starter points* and *Junction Points* in the given *correlation area*. In order to find the *major starter points*, two techniques can be used [8]. In both of these techniques, the pixels in the given *skeleton area* are processed row-wise.

F Traversal through the correlation area

Definition 5: The *traversal direction* is the direction from the current pixel to the next pixel to be visited during the traversal.
Definition 6: An *end point* is a pixel point in the correlation area, in which there is no neighboring pixel to visit next

After finding all the *major starter points*, and the *junction points*, the algorithm starts traversing (hereafter referred to as initial splitting) through the connected character skeleton, starting from the *major starter point* of the list of major starter points. In this initial splitting, the segments are identified in the traversal path based on the junction point list. Moreover, the *minor starter points* are also identified at each junction of the skeleton and they are queued to a different queue, which is hereafter referred to as *minor_starters*. Once the traversal reaches a *junction point*, or an *end point*, which is a pixel point with no neighboring pixel to visit next, the focus is shifted to the identified *minor starter points* in the *minor_starters* queue. Then the algorithm starts traversing the unvisited paths of the skeleton by starting with each *minor starter point* in the *minor_starters* queue. In these traversals, the algorithm also segments the path that is being visited up to *junction point* or an *end point* into initial segments.

The process mentioned above is continued with all the unvisited *major starter points* in the *major_starters* queue, until all the unvisited paths in the *correlation area* are visited. The risk of visiting the same pixel (hence the same path) more than once during each splitting traversal is eliminated by memorizing all the visited pixel points and only visiting the unvisited pixels in the later traversals. Therefore it is guaranteed that a path in the skeleton is visited only once and hence the same segment is not identified twice.
The initial splitter routine is as follows.

```
Function init_split(major_starter_point) returns segments
  Begin
    current_segment = empty // Segment points, current segment
    current_point = empty // Point, refers to the current pixel point
    current_direction = empty // String, current traversal direction.
    next_point = empty // Point, next pixel point to be visited
    segments = empty // segments array, identified segments
    minor_starters = empty // Point queue, minor starter points

    while (there are more points in the minor_starters queue OR
                    current_point is nonempty) do
    if(current_point =empty ) then
    current_point = minor_starters.deque()
    initialize the current_segment
    if(current_point is unvisited) then
    current_segment.add(current_point)
    make the current_point as visited
    end if end if
if(unvisited eight adjacent neighbors of current_point exist) then
      neighbors =get all unvisited adjacent neighbors of
current_point
      if(current_point    ==junction_point)
minor_starters.enque(neighbors) // neighbors at the   junction
segment.add(current_segment) // segmentation at the junction
    current_point= empty
else next_point = choose any neighbor of the current_point
    current_segment.add(next_point)
    mark next_point as visited
    current_direction = get the current traversal direction
    current_point = next_point
    end if end if
else // if there are no unvisited neighbors to visit
    segment.add(current_segment)
    current_point = empty
    end if
    end while
return segments
End
```

The initial splitter process mentioned above is continued with all the unvisited *major starter points* in the *major_starters* queue, and the final output of the segmentation is used as the input to the complete splitting process, which produces complete segmentation of the connected character skeleton. The complete splitter process is used to analyze each initial segment to split each of the candidate segments in to complete character segments.
The complete splitter routine is as follows.

```
Function complete_split(init_segments) returns all_segments
  Begin
    all_segments = empty // Segment queue,  all identified segments
    current_segment = empty // selected initial segment
    tmp_segment = empty // segment to hold next 5 points
    current_point = empty // Point, refers to the current pixel point
    next_point = empty // Point, next pixel point to be visited
    start_point = empty // Point, first point of  initial segment
    end_point = empty // Point, last point of a selected initial segment
    direction = empty // String, to hold the traversal direction.

    current_point = start_point
    current_segment.add(current_point)
    end_point=get end point of the selected initial segment

while (no termination occur) do
    if(current_segment >1 ) then
```

```
        next_point = get next segment point of the current
point(init_segment)
        if(does neighbour in the same direction) then
                current_segment.add(next_point)
                current_point=next_point
                if(current_point is the end point)
                terminate condition occurs
                all_segments.add(current_segment)
        end if
        else // neighbour is not in the same direction
                //I.e. the traversal direction changes.
                 tmp_segment = get next 5 pixels in the path.
        if(IsAbruptChange(current_segment, tmp_segment)) then
                all_segment.add(current_segment)
                current_segment = tmp_segment
                if(current_point is the end point)
                terminate condition occurs
                all_segments.add(current_segment)
                end if
         else
            // the traversal can continue with the same segment.
      add all the points in the tmp_segment to current_segment
        if(current_point is the end point)
                terminate condition occurs
                all_segments.add(current_segment)
        end if end if end if
        else
        next_point = get next segment point of the current
        point(init_segment)
        current_segment.add(next_point)
        current_direction = get the current traversal direction
        current_point=next_point
        end if
        end while
 return all_segments
End
```

G Getting Next 5 Pixel Points in the Segment

The intention of getting next 5 pixel points into a separate data structure refereed to as *tmp_segment*, is to find the new *written direction*. The segmentation decision is based on the abrupt change in the *written direction* [8].

Definition 7: The w*ritten direction* is the direction of a particular sequence of pixels to which they were written.

According to the complete segmentation algorithm, as long as the *current traversal direction* remains unchanged (if it can find an unvisited neighboring pixel in the *current traversal direction*), the algorithm considers that the path, which is being visited, belongs to the same segment. If the *current traversal direction* changes, then the algorithm goes and checks for an abrupt change in *written direction*. In this work, calculation of *written direction* of a sequence of pixels is implemented as proposed in [8]. Furthermore, the experimental rule base proposed in [7] is used to segment handwritten uppercase English characters in to meaningful segments. In addition, the current traversal direction in [8] is used to find out the treaversal direction from the current pixel point to the next point in the connected character skeleton.

H Merging of Segments across junctions

After the segmentation process and noise removal [1], each segment can be considered as totally spitted segments of the connected character skeleton. With respect to the work proposed in [1], [7] and [8], for the recognition, each of the isolated character should undergo proper segmentation, in order to obtain meaningful segments as the output.

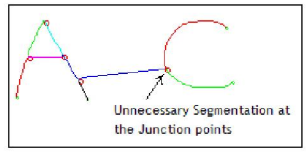

Fig. 7. *Unnecessary segmentation at a junction point*

When the consideration is only focused on isolated characters, it is rather easy to do the segmentation in to meaningful skeletons as described in the work of [8]. In this research, not only the individual character segments, but also the connection between each individual character has to be considered for the segmentation in order to get the segments out from the connected character input. This may lead to unnecessary segmentation across certain connection as described in the Fig. 7. Therefore, Merging of each segment across junction is proposed, to avoid unnecessary segmentation and to reconstruct meaningful segments.

In this research, each of the segments in the complete segment list was named with respect to its participating junction point number(s).

Definition 8: The *junction point number* is a unique integer assigned to the identified junction points of the *correlation area*. As an example, correlation area with 6 junction points, each point can be named by the integers starting from 1 to 6 in the same order in which each point was identified.
As shown in the Fig. 8, in this method each of the segment objects can be composed in to a first junction number, second junction number and a pixel point segment. The Junction number (first or second) depends on the direction of segmentation. In this work, the first coordinate point of a segment is considered as the first junction number and last coordinates point as the second junction number. Any segment with a corner point, which is not a junction point is assigned zero as the junction number at that corner.

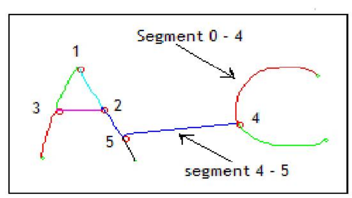

Fig. 8. Junction based naming process

Ex: Segment (first Junction no - second junction no)

Segment (3 – 0)

In the process of Merging, each segment in the total segment is categorized in accordance with the related junction number. As an example, junction point number 3 is used to create a category, which consists of all segments, which have a junction point number 3 at first or second junction number. This method is used to create groups of segments with respect to all the junction point numbers in the correlation area except the points, which have number zero.

In this work, each of the group is separately examined to select possible merging at each junction point number. In a particular junction point number, each segment is analyzed with all the other participating segments within the group. At the analyzing time, each pairs of segments within the group, are used to merge in order to get merged segments. Each identified segment was then said to possess a certain number of characteristics, for each a fuzzy value was calculated, using the methods described in [1].

According to the calculated fuzzy values of a particular segment, an experimental rule base is used to extract meaningful segment within a particular group. Each identified merged segments is mapped and replaced with a segment, if the segment is a participant in the merging is recorded in other groups as well (if one or both of the participating segments in the merged segment).

All successfully merged meaningful segments are recorded in a group called *major segment group*, and all the others not merged are traced in to a separate structure called *minor segment group*. In this method, it was found that the minor segment group could isolate an unnecessary connection between individual characters. In this work, each of the segments in the major segment group is used to formulate new input character image by selecting segments from the minor segment group. This method is used to create input character image with several correlation areas. For the recognition, the work proposed in [1], is used to identify each individual character by sending each correlation area to the isolated character identification process. In this work, the correlation area, which only produces successful two characters out put, is considered as the separated characters output of the original connected character input.

III EXPERIMENTAL RESULTS

The algorithm was tested with the connected character skeletons obtained by the skeletonization process, of handwritten upper case English characters. Fig. 9, Shows the complete segmentations of the connected character skeleton after applying the initial splitter and the complete splitter.

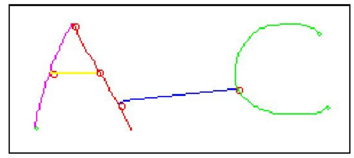

Fig. 9. Total meaningful segments from merging

At the character isolation stage, all merged segments (major segments) are combined with each of minor segments to generate set of different connection sketches (Fig. 10,). These sketches are used to formulate new input character image to an individual character recognition system, in order to recognize characters of the connection string.

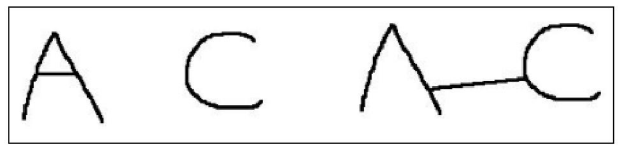

Fig. 10. Connection sketches for the recognition process

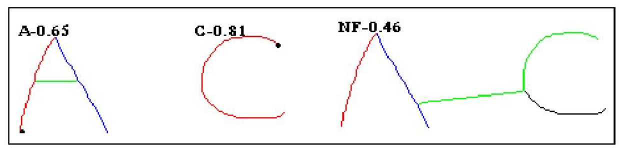

Fig. 11. Final output from the recognition system

IV CONCLUSIONS & RECOMMENDATIONS

In the work presented above, it was found that the proposed algorithm for the connected character separation was an extremely reliable and relatively simple method for generating the fuzzy description of connected character patterns, once the characters were properly segmented. The segmentation algorithm developed for this work was capable of separating the connections in the uppercase English handwritten characters into meaningful segments accurately. The next step in the development of this system will be a connected character separation algorithm in lowercase handwritten English letters and numeric.

The recognition rate was found to be 100%, when the same characters that were used for the training, fed to the system for identification with a connection in between. Therefore, it can be concluded that the proposed method is 100% accurate for machine printed characters, which have unwanted connection. It was observed that it was very flexible and simple to implement, compared to other available methods for offline handwritten connected character recognition.

Although the method used in this work does not deal with the two stroke connections from two adjacent digits touching end to end, the system can still be used for the separation of any handwritten connected characters with single or double connections. As a future development it can also be suggested a development of an identification method, which is capable of separating the above mentioned connections. The whole method suggested in this paper can be applicable to any handwritten or machine written offline character recognition system for any character set in any language. The main requirements would be, a different connected character separation algorithm for different character sets and a proper rule base for the meaningful segmentation.

REFERENCES

[1] K. B. M. R. Batuwita, G.E.M.D.C. Bandara (2005), "*An online adaptable fuzzy system for offline handwritten Character recognition*", Proceedings of 11th World Congress of International Fuzzy Systems Association (IFSA 2005), Beijing, China, 2005, Springer-Tsinghua, Vol. II, p.1185-1190

[2] LI Guo-hong, SHI Peng-fei (2003), "*An approach to offline handwritten Chinese character recognition based on segment evaluation of adaptive duration* ", Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, Shanghai, China.

[3] Zongkang Lu, Zheru Chi, Wan- Chi Siu, Pengfei Shi, "*A Background thinning based approach for separating and recognizing connected handwritten digit strings*", Department of Electronic Engineering , the Hong Kong Polytechnic University, Hong Kong, Institute of Pattern Recognition & Image Processing, Shanghai Jiaotong University, China

[4] Akihiro Nomura, Kazuyuki Michishita, Seiichi Uchida, and Masakazu Suzuki, "Detection and Segmentation of Touching Characters in Mathematical Expressions", Graduate School of Mathematics, Faculty of Information Science and Electrical Engineering, Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka-shi, 812-8581 Japan.

[5] Daekeun Y. , Gyeonghwan K., "*An approach for locating segmentation points of handwritten digit strings using a neural network* ", Dept. of Electronic Engineering Sogang University, CPO Box 1142, Seoul 100-611 Korea.

[6] Yi Lu, Beverly Haist, Laurel Harmon, Jhon Trendkle, Robert Vogt., "*An Accurate and Efficient System for Segmenting Machine-Printed Text* ", Environmental Research Institute of Michigan, Ann Arbor, MI .

[7]. K. B. M. R. Batuwita, G.E.M.D.C. Bandara (2005), "*An Improved Segmentation Algorithm for Individual Offline Handwritten Character Segmentation*", Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'2005), November 28-30, 2005, Vienna, Austria.

[8]. K. B. M. R. Batuwita, G.E.M.D.C. Bandara (2005), "*New Segmentation Algorithm for Individual Offline Handwritten Character Segmentation*", Proceedings of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'2005), August, 28-30, Changsha, China.