



**CONNECTED CHARACTER SEGMENTATION FOR
OFFLINE HANDWRITTEN CHARACTER
RECOGNITION**

by

U.K.S. Jayarathna (PS/01/122)

Department of Statistics and Computer Science
Faculty of Science

University of Peradeniya

June 2006

This dissertation is submitted in partial fulfillment of the requirement of the Degree of Bachelor of Science in Computer Science (Special) of the Faculty of Science, University of Peradeniya

DECLARATION

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a Degree or Diploma in any University and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due references is made in this text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for inter-library loans, and for the title and summary to be made available to outside organizations.

Signature of Candidate

Date: 06th of June 2006

.....
U.K.S. Jayarathna (PS/01/122)
Department of Stat. & Computer Science
Faculty of Science
University of Peradeniya

Supervisory Committee

Signature of Supervisor (Chair)

Date:.....

.....
Dr. G.E.M.D.C. Bandara
Department of Production Engineering
Faculty of Engineering
University of Peradeniya

Date:.....

.....
Dr. S.R. Kodithuwakku
Department of Stat. & Computer Science
Faculty of Science
University of Peradeniya

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, Binarization , Fuzzy Rules, Connected digit Strings, Segmentation

ACKNOWLEDGEMENT

First and foremost, I wish to express my heartfelt thanks and appreciation to my supervisor Dr. G.E.M.D.C Bandara, Senior Lecturer of the Department of Production Engineering, for his continuous support, guidance and supervision throughout the project. A great part of the project's success is due to his help and guidance, without which this project would not be possible.

I also wish to express my gratitude to Dr. S.R. Kodithuwakku, Senior Lecturer, Department of Stat. & Computer Science, Mr. Romesh Ranawana, Computing Laboratory, University of Oxford, United Kingdom, Mr. Ruckshan Batuwita, Assistant Lecturer, Department of Stat. & Computer Science, for the support they provide for me to carryout this task.

I wish to thank specially to my Family and Nim for the love, support and encouragement given to me to do my best.

My heartfelt thanks goes to my dearest friends Jayantha Kumara and Indika Wimalasuriya for constantly giving me the courage, support and the technical know-how throughout the year.

Last but not least I would like to convey my gratitude to everyone who helped me during the project in everyway, whose names go unmentioned.

TABLE OF CONTENTS

ABSTRACT	III
ACKNOWLEDGEMENT	IV
TABLE OF CONTENTS	V
LIST OF FIGURES	VI
LIST OF TABLES	VII
ACRONYMS	VIII
CHAPTER 1	2
1.1 INTRODUCTION.....	2
1.2 CASE FOR THE NEED OF THE PROJECT	2
1.3 CHAPTER BREAKDOWN	3
CHAPTER 2	6
2.1 BACKGROUND.....	6
CHAPTER 3	9
3.1 METHODOLOGY	9
3.2 PREPROCESSING	10
3.2.1 BINARIZATION	10
3.2.2 SKELETONIZATION.....	11
3.2.3 ISOLATION OF CONNECTED CHARACTER SKELETON	11
3.3 JUNCTION BASED SEGMENTATION	12
3.4 THE ALGORITHM	14
3.4.1 IDENTIFICATION OF STARTER/JUNCTION POINTS	16
3.4.2 TRAVERSAL THROUGH THE CORRELATION AREA	16
3.4.3 IDENTIFICATION OF MINOR STARTER POINTS.....	18
3.4.4 GETTING NEXT 5 PIXEL POINTS IN THE SEGMENT.....	20
3.4.5 DETERMINATION OF CURRENT TRAVERSAL DIRECTION	20
3.4.6 NOISE REMOVAL.....	21
3.5 MERGING OF SEGMENTS ACROSS JUNCTIONS	21
3.5.1 CHARACTER SKETCH ISOLATION & NEW INPUT IMAGE.....	23
3.6 INDIVIDUAL CHARACTER RECOGNITION	24
3.6.1 CALCULATION OF FUZZY FEATURES	25
3.6.2 RELATIVE POSITION.....	25
3.6.3 GEOMETRICAL FEATURE DETECTION.....	26
3.6.4 CALCULATION OF LINE TYPES AND RELATIVE LENGTHS	27
3.6.5 CALCULATION OF CURVE TYPES	28
3.7 THE DATABASE DESIGN	31
3.7.1 THE TRAINING PROCESS	32
3.7.2 THE RECOGNITION PROCESS	33
CHAPTER 4	36
4.1 RESULTS AND DISCUSSION.....	36
CHAPTER 5	40
5.1 CONCLUSIONS AND RECOMMENDATIONS	40
REFERENCES	41

LIST OF FIGURES

Fig. 1. Connected Character String.....	2
Fig. 2. Different connection styles	9
Fig. 3. Block diagram of the proposed system.....	10
Fig. 4. Binarized image of a connected handwritten characters	11
Fig. 5. Skeletonized image of a connected handwritten characters	11
Fig. 6. Representation of the <i>correlation area</i> of a connected two-digit string.....	12
Fig. 7. <i>Junction points</i> of the correlation area	13
Fig. 8. Initial Segmentation of the correlation area	14
Fig. 9. Complete segmentation in to meaningful segments	14
Fig. 10. Identification of <i>minor starter</i> points	18
Fig. 11. Unnecessary segmentation at a junction point	21
Fig. 12. Junction based naming process.....	22
Fig. 13. Block diagram of the individual character recognition system	24
Fig. 14. Isolated individual characters for the recognition input	24
Fig. 15. Character skeleton 'A' and its <i>universe_of_discourse</i>	25
Fig. 16. Relative Position with respect to the universe of discourse	25
Fig. 17. Width, height and slant length of the universe of discourse.....	28
Fig. 18. The database design.....	31
Fig. 19. Membership functions to describe the characteristics of each segments	32
Fig. 20. Total meaningful segments from the merging process.....	36
Fig. 21. Character set with which the system was tested.....	36
Fig. 22. Set of auto generated different <i>Connection sketches</i>	37
Fig. 23. Total meaningful segmentation of individual characters	37
Fig. 24. Partial Segment table entries for character 'A' in Fig. 33.....	37
Fig. 25. Partial Segment table entries for character 'C' in Fig. 33	37
Fig. 26. Final output from the recognition system.....	37

LIST OF TABLES

Table. 1. Types of meaningful straight lines and meaningful arcs that a character skeleton should be broken into	13
Table. 2. Types of points identified by the algorithm.....	16
Table. 3. Eight adjacent neighbors of the current pixel (i, j) and Determination of the current traversal direction	21
Table. 4. Set of fuzzy features, which were calculated for each segment	23
Table. 5. The equality fuzzy values generated by the system after comparing the features of the segments of the given character (Fig. 20) with the features of segments of the characters in the database having the identical number of segments (in this case 3 segments) as the given character.....	38
Table 6. The equality fuzzy values generated by the system after comparing the features of the segments of the given character 'C' (Fig. 20) with the features of segments of the characters in the database having the identical number of segments (in this case 1 segment) as the given character.	38

ACRONYMS

ICR	: Intelligent Character Recognition
OCR	: Optical Character Recognition



CHAPTER 1



1 Preamble

- 1.1. Introduction
- 1.2. Case for the need of the project
- 1.3. Chapter Breakdown

CHAPTER 1

1.1 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. Character recognition with better accuracy is important, to make these features effective and efficient.



Fig. 1. Connected Character String

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 (Fig. 1), 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. However, connection between digits strings in handwritten scripts is common.

1.2 CASE FOR THE NEED OF THE PROJECT

Issues that related to the segmentation and recognition of handwritten strings have been dealt in various ways for the several decades. It is well known that the difficulties stem from not only variations existing in shape but also from various ways of touching. And, in many cases, the segmentation and the recognition could not be considered independently, when connection between characters is assumed. It is desirable to develop segmentation techniques, which are style independent. However, it is also reasonable to develop methods which are specialized for handling broad

categories of character segmentation situations [4], such as those listed below (the order of increasing difficulty).

- Uniformly spaced characters, i.e. fixed-pitch fonts.
- Well-separated and unbroken characters, not uniformly spaced
- Broken characters
- Connected characters
- Broken and Connected characters
- Broken and italic characters
- Connected and italic characters
- Script characters

1.3 CHAPTER BREAKDOWN

The initial part of this dissertation includes a detailed discussion on the handwritten character recognition and how they are related with the offline and online dilemma. It provides details on the already available methods to solve the connected character segmentation and as well as other aspects of the offline handwritten character recognition.

The second section of the dissertation discusses the methodology used in this research in order to recognize connected handwritten characters. It discusses its critical issues in the initial processing, how the algorithm works and its detailed implementation issues.

The final section of this report discusses the experimental results using the proposed methodology, its critical issues and feasibility.

Following is a brief depiction about the forth-coming chapters and their contents.

CHAPTER 2: BACKGROUND

This chapter describes the background information of the connected character recognition and literature reviews on each identified existing technologies and weigh against each choice.

CHAPTER 3: METHODOLOGY

This chapter consists of the methodology used in this research. Further it discusses the algorithmic approach for the system realization.

CHAPTER 4: RESULTS & DISCUSSION

This chapter describes the experimental results acquired from the research

CHAPTER 5: CONCLUSIONS & RECOMMENDATIONS

This chapter is used to critically discuss the results of the project and how far the objectives have been satisfied. It also suggests any predictable improvements to the research.

REFERENCES:

Referred list of books, journals, research papers and web sites from the related work.



CHAPTER
2



2 Background
2.1 Background

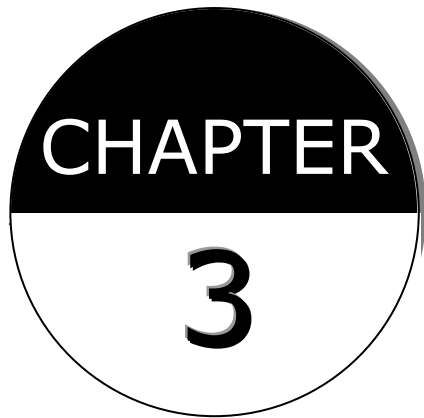
CHAPTER 2

2.1 BACKGROUND

Handwriting recognition technology is steadily growing toward its maturity. Significant results have been achieved in the past few years both in on-line and off-line handwriting recognition. While generic content text recognition seems to be a long-term goal, some less ambitious tasks are currently being investigated that address relevant problems such as the recognition of postal addresses, and the legal amount on bank cheques. Current systems are capable of transcribing handwriting with average recognition rates of 90–99%, depending on the constraints imposed (e.g. connected or touched characters, size of the vocabulary, writer-dependence, writing style, etc.), and also on the experimental conditions [7]. One of the most common constraints of current recognition systems is recognizing the connected or touched character strings that are present in an off-line handwritten paradigm.

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.



CHAPTER

3

3 Methodology

- 3.1 Methodology
- 3.2 Preprocessing
- 3.3 Junction Based Segmentation
- 3.4 The algorithm
- 3.5 Merging of Segments across Junction
- 3.6 Individual Character Recognition
- 3.7 The Database Design

CHAPTER 3

3.1 METHODOLOGY

This section discusses the methodology used in this research. When considering existing systems, there are some major features defined in [3], which, a connection of adjacent handwritten digits can be categorized into (**Fig. 2**),

- (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.

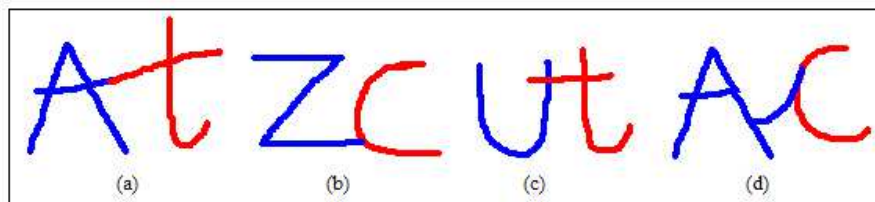


Fig. 2. Different connection styles

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 Stroked 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 proposed system operates in two modes, namely, Character Isolation mode and the Recognition mode. The block diagram of the proposed system is depicted in **Fig. 3**.

This paper will cover the proposed system by presenting the preprocessing of the connected character image, namely, the binarization, the skeletonization, isolation of connected characters, Junction based segmentation, respectively. Merging of the segments across junctions by calculating the fuzzy characteristic values for each of the merged segment is described in section (3.5).

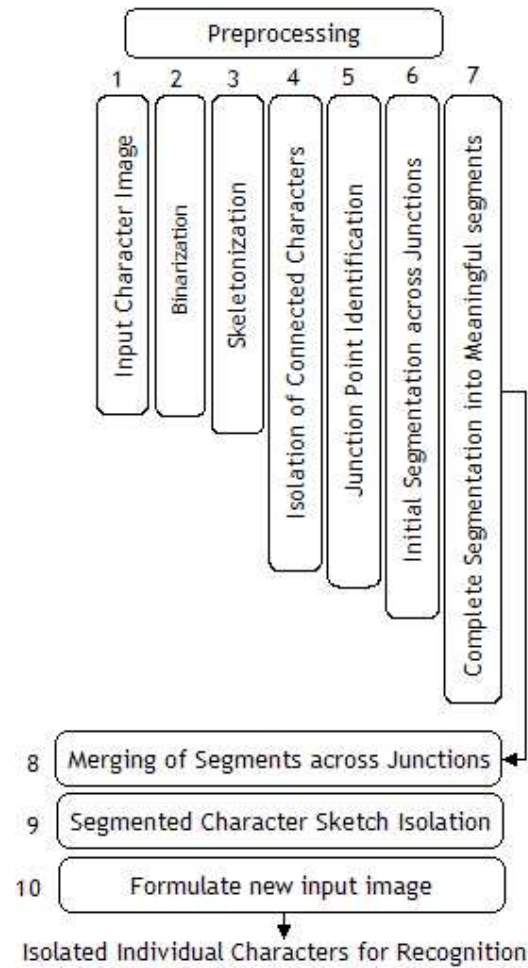


Fig. 3. Block diagram of the proposed system

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]. Segmented Character Sketch isolation and the Formulation of the new Input character image (isolated individual characters) are described in section (3.5.1). 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 3.6). Section 4 outlines some experimental results. The paper concludes with conclusions and future work outlines.

3.2 PREPROCESSING

3.2.1 BINARIZATION

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. Otherwise the loss of information and/or noises would result in later processing phases due to various intensity values of the pixels. 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 correctly binarized image is shown in Fig. 4.

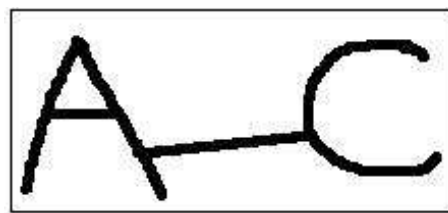


Fig. 4. Binarized image of a connected handwritten characters

3.2.2 SKELETONIZATION

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. The result of the skeletonization process obtained by applying [8] is depicted in **Fig. 5**.

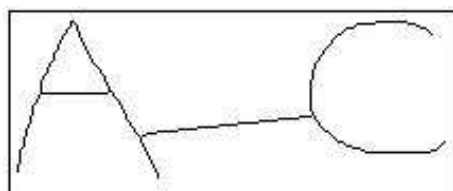


Fig. 5. Skeletonized image of a connected handwritten characters

3.2.3 ISOLATION OF CONNECTED CHARACTER SKELETON

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 *correlation area*. For the individual characters; the method proposed in [1] is used. 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 skeletons 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*.

Definition 1: A *correlation area* is a rectangular area in the image, which contains connected character skeleton. **Fig. 6.** depicts how a correlation area can be represented.

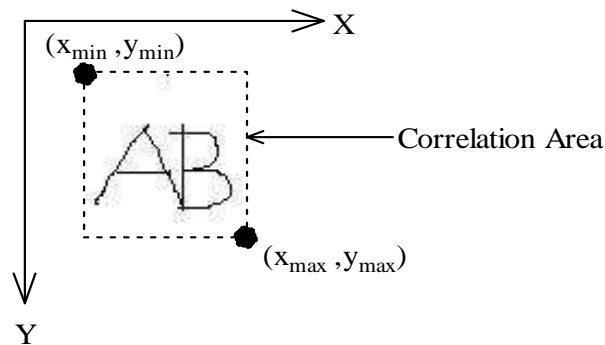


Fig. 6. Representation of the *correlation area* of a connected two-digit string

3.3 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. The proposed segmentation algorithm should produce a set of segments into, which a character skeleton is broken. 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 **Table. 1.** but not an arbitrary segment.

Table. 1. Types of meaningful straight lines and meaningful arcs that a character skeleton should be broken into

Meaningful Straight Lines		Meaningful Arcs	
Horizontal Line	—	C like	⤿
Vertical Line		D like	⤵
Positive Slanted	↘	U like	⤶
Negative Slanted	↗	A like	⤿
		O like	○

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. 7**) in the correlation area, each is used to segment the connected character skeleton in to initial segments.

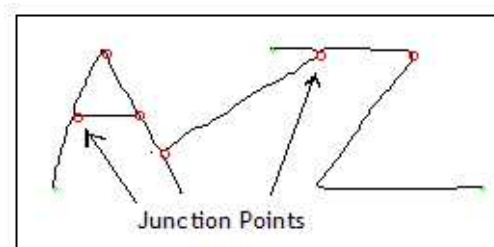


Fig. 7. *Junction points* of the correlation area

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. 8**) would not be segmented into a “negative slanted” and two “Horizontal Lines” because of the non-Junction point connection in between.

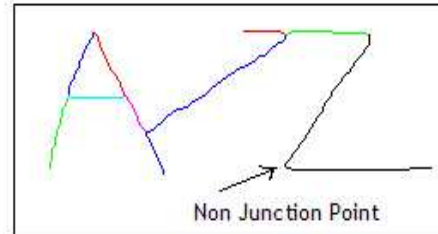


Fig. 8. Initial Segmentation of the correlation area

To compensate for this, a separate segmentation algorithm is used with the rule based segmentation approach proposed in [7] (**Fig. 9**).

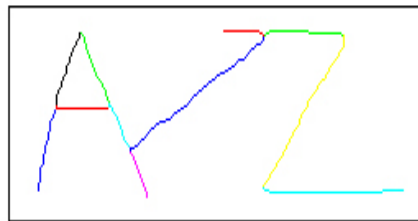


Fig. 9. Complete segmentation in to meaningful segments

3.4 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 *Segment*, 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 3.4.1.

Definition 5: A *minor starter point* is a *starter point*, which is identified during the traversal through the skeleton. The identification of *minor starter points* is described in Section 3.4.3.

The main routing of the connected characters segmentation algorithm (hereafter referred 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 to store major starter points.

`junction_points = empty` // a queue of Point to store junction points.

`partial_Segment = empty` / an array of Segment, to store the partial segments identified by starting traversal with a particular major starter point.*/

`init_Segments = empty` // an array of Segment, to store all the identified partial segments

`complete_Segments = empty` /* an array of Segment, to store complete character segments */

`total_Segments = empty` /* an array of Segment, to store completed , named character 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

3.4.1 IDENTIFICATION OF STARTER/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.

Table. 2. Types of points identified by the algorithm

Point Type	Identification Method
Major Starter Point (Technique 1)	All the pixel points in the <i>correlation area</i> , those having only one neighboring pixel
Major Starter Point (Technique 2)	In case of character ‘O’ and number zero, they cannot be obtained by the Technique 1, since all the points in such a one pixel thickness, closed ‘O’ like curve would have at least two neighboring pixels. In such a case, the first pixel, which is found on the character skeleton, is taken as its <i>major starter point</i> .
Junction Point	All the pixel points in the <i>correlation area</i> , those having three or more neighboring pixels

3.4.2 TRAVERSAL THROUGH THE CORRELATION AREA

Definition 6: The *traversal direction* is the direction from the current pixel to the next pixel to be visited during the traversal. The determination of *current traversal direction* is described in Section 3.4.4.

Definition 7: 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*. (The identification of *minor starter points* is discussed in section 3.4.3). 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 to store points of the current segment
current_point = empty // Point, refers to the current pixel point
current_direction = empty // String, to hold the current traversal direction.
next_point = empty // Point, refers to the next pixel point to be visited
segments = empty // segments array, to hold the identified segments
minor_starters = empty // Point queue, to store minor starter points
minor_starters.enqueue(major_starter_point)
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.dequeue()
        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 the current_point
        if(current_point ==junction_point)
            minor_starters.enqueue(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
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

```

3.4.3 IDENTIFICATION OF MINOR STARTER POINTS

Let us consider the traversal through the character skeleton 'B' in **Fig. 10**. The traversal starts with the *major starter point* 'a' and continues to the junction 'J₁'. At this junction, the current pixel point has two unvisited neighbors. Since there is a neighboring pixel in to the *current traversal direction*, the algorithm chooses that pixel (n₁) between the two neighboring pixels as the next pixel point to visit. It is clear that the other neighboring pixel (n₂) is a starter point of another path in the skeleton. Therefore the point n₂ is identified as a *minor starter point* and inserted into the *minor_starters* queue for later consideration. At every junction in the skeleton, there may be one or more *minor starter points* identified.

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.

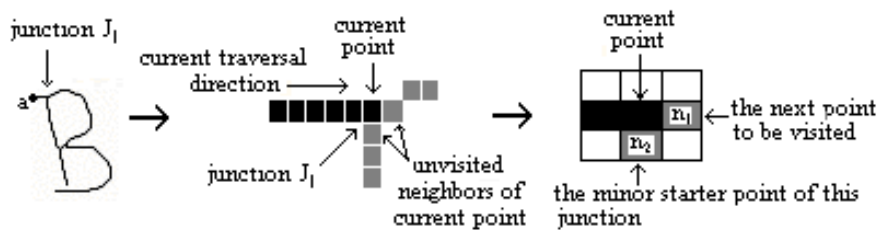


Fig. 10. Identification of *minor starter points*

The complete splitter routine is as follows.

Function complete_split(init_segments) returns all_segments

Begin

```

all_segments = empty // Segment queue, to store all identified segments
current_segment = empty // selected initial segment to be consider
tmp_segment = empty // segment to hold next 5 points

```

```

current_point = empty // Point, refers to the current pixel point
next_point = empty // Point, refers to the next pixel point to be visited
start_point = empty // Point, first point of a selected initial segment
end_point = empty // Point, last point of a selected initial segment
direction = empty // String, to hold the traversal direction.

start_point =get first point of the selected initial segment
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

```

3.4.4 GETTING NEXT 5 PIXEL POINTS IN THE SEGMENT

The intention of getting next 5 pixel points into a separate data structure referred to as *tmp_segment*, is to find the new *written direction* as described in definition 7. Getting next 5 pixel points in the path is carried out as follows.

```
tmp_segment .add(next_point)
curr_point=next_point
for each of the segment in the init_segment do
  if(curr_point==point in init_segment)
    count
  end if
end for
while(tmp_segment <5 and init_segment is greater than the count+1) do
  next_point=get the next point in the init_segment
  tem_segment.add(next_point)
  current_direction = get the current traversal direction
  curr_point=next_point
  increment count
end while
```

The segmentation decision is based on the abrupt change in the *written direction* [8].

Definition 8: The *written 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 into meaningful segments.

3.4.5 DETERMINATION OF CURRENT TRAVERSAL DIRECTION

Let us consider all the eight adjacent neighbors of a current pixel (i, j). The *current traversal direction* can be defined as described in **Table. 3.** according to the neighboring pixel, which is chosen as the next pixel to be visited.

Table. 3. Eight adjacent neighbors of the current pixel (i, j) and Determination of the current traversal direction

$(i-1, j-1)$ LU (LEFT_UP)	$(i, j-1)$ U (UP)	$(i+1, j-1)$ RU (RIGHT_UP)
$(i-1, j)$ L (LEFT)	(i, j)	$(i+1, j)$ R (RIGHT)
$(i-1, j+1)$ LD (LEFT_DOWN)	$(j, j+1)$ D (DOWN)	$(i+1, j+1)$ RD (RIGHT_DOWN)

3.4.6 NOISE REMOVAL

In the process of noise removal, the segments resulted due to the noises in the *correlation area* are removed. The average size of the character skeletons considered in this research is 50*50 pixels. Accordingly, the minimum size of the segment was taken as 5 pixel points. Therefore under the noise removal, the segments, which contain less than 5 pixel points, are discarded from the set of resulted segments.

3.5 MERGING OF SEGMENTS ACROSS JUNCTIONS

After the segmentation process, 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.

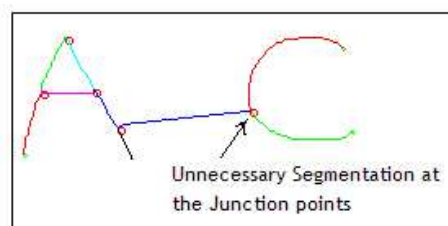


Fig. 11. Unnecessary segmentation at a junction point

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 stroked 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. 11**. 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 9: 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. 12**.in this method each of the segment object 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.

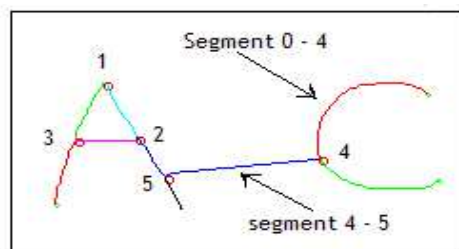


Fig. 12. Junction based naming process

So, a segment with a first junction and the second junction can be named as follows

“Segment first Junction no - second junction no”

Ex: 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.

3.5.1 CHARACTER SKETCH ISOLATION & NEW INPUT IMAGE

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 (**Table. 4**), for each a fuzzy value was calculated, using the methods described in [1].

Table. 4. Set of fuzzy features, which were calculated for each segment

<i>Relative Positions</i>	<i>Straightness & Arc-ness</i>
MHP(μ Horizontal position)	MSTR(μ Straightness)
MVP(μ Vertical position)	MARC(μ Arc-ness)
<i>Line Types</i>	<i>Curve Types</i>
MVL(μ Vertical line)	MCL(μ C-like)
MHL(μ Horizontal line)	MDL(μ D-like)
MPS(μ Positive slant (/))	MAL(μ A like)
MNS(μ Negative slant (\))	MUL(μ U-like)
<i>Relative Lengths</i>	MOL(μ O-like)
MLENH(μ Length Horizontal)	
MLENV(μ Length Vertical)	
MLENS(μ Length Slant)	

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 stroked 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.

3.6 INDIVIDUAL CHARACTER RECOGNITION

This section discusses the individual character recognition used in this research [1]. The system operates in two modes, namely, the training mode and the recognition mode. The block diagram of the system is depicted in **Fig. 13**.

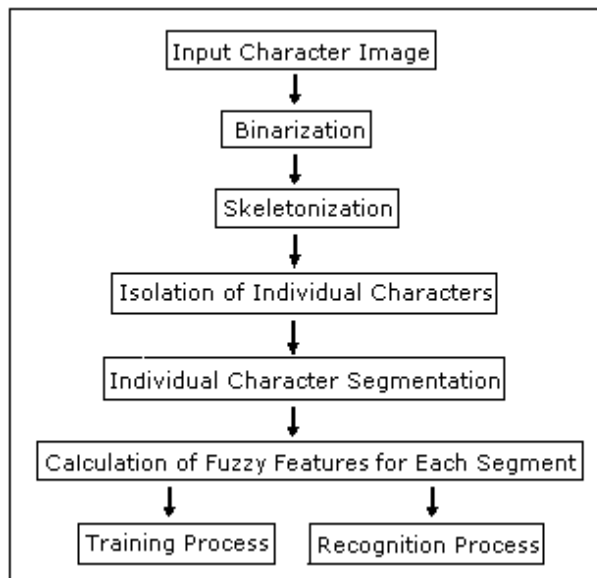


Fig. 13. Block diagram of the individual character recognition system

Once the separate segments of a given character skeleton have been properly identified (as shown in **Fig. 14**), the individual characteristics of each resulted segment can be calculated.



Fig. 14. Isolated individual characters for the recognition input

3.6.1 CALCULATION OF FUZZY FEATURES

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] and [7]. Each of this fuzzy value was calculated with respect to the *universe_of_discourse*¹ of the particular character skeleton (Fig. 15).

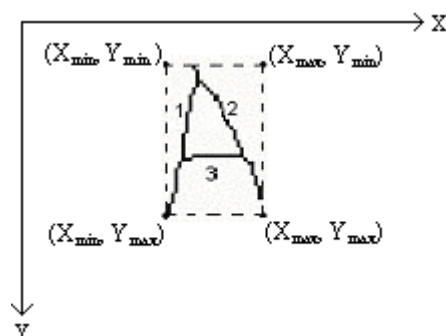


Fig. 15. Character skeleton 'A' and its *universe_of_discourse*

The following fuzzy features were calculated using the same method used in [1] and [7] for isolated offline character recognition.

3.6.2 RELATIVE POSITION

The relative position of a given segment with respect to the *universe_of_discourse* can be determined as follows.

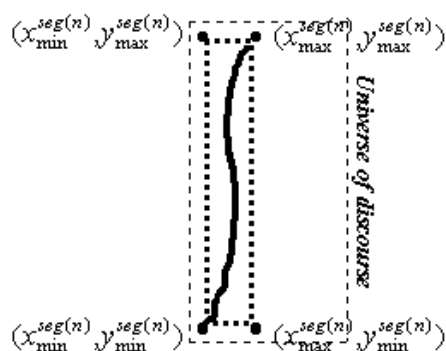


Fig. 16. Relative Position with respect to the universe of discourse

Then the coordinates of the center points are calculated as follows:

¹The smallest rectangular area that a character skeleton fits into.

$$x_{CENTER}^{seg(n)} = \frac{x_{min}^{seg(n)} + x_{max}^{seg(n)}}{2}$$

$$y_{CENTER}^{seg(n)} = \frac{y_{min}^{seg(n)} + y_{max}^{seg(n)}}{2}$$

The relative position of the given segment is then expressed as follows:

$$\mu_{HP}^{seg(n)} = \frac{x_{CENTER}^{seg(n)} - x_{min}^{seg(n)}}{x_{max}^{seg(n)} - x_{min}^{seg(n)}}$$

$$\mu_{VP}^{seg(n)} = \frac{y_{CENTER}^{seg(n)} - y_{min}^{seg(n)}}{y_{max}^{seg(n)} - y_{min}^{seg(n)}}$$

Where the terms HP and VP stand for ‘Horizontal Position’ and ‘Vertical Position’ respectively.

3.6.3 GEOMETRICAL FEATURE DETECTION

A given segment is said to be either an ‘arc’ or a ‘straight-line’. The fuzzy values associated with these features are ‘arc-ness’ and ‘straightness’. These two values are complementary.

$$\mu_{ARC-NESS}^{seg(n)} + \mu_{STRAIGHTNESS}^{seg(n)} = 1$$

The arc-ness and the straightness can be calculated by the use of the following two expressions:

$$\mu_{STRAIGHTNESS}^{seg(n)} = \frac{d_{p_0 p_N}^{seg(n)}}{\sum_{k=0}^{N-1} d_{p_k p_{k+1}}^{seg(n)}}$$

$$\mu_{ARC-NESS}^{seg(n)} = 1 - \frac{d_{p_0 p_N}^{seg(n)}}{\sum_{k=0}^{N-1} d_{p_k p_{k+1}}^{seg(n)}}$$

Where, $d_{p_k p_{k+1}}^{seg(n)}$ stands for the straight-line distance between point k and point (k+1) on the nth segment. N depicts the number of elements in the segment. A threshold value (e.g. 0.6) could be used to determine whether the given segment is a straight line or an arc.

3.6.4 CALCULATION OF LINE TYPES AND RELATIVE LENGTHS

If the given segment is determined to be a straight line, then the line type can be calculated using the following equations.

$$\begin{aligned} \text{MVL} &= \mu_{VL}^{seg(n)} = \text{MAX}(\Lambda(\theta^{seg(n)}, 90, 90), \Lambda(\theta^{seg(n)}, 90, 270)) \\ \text{MHL} &= \mu_{HL}^{seg(n)} = \text{MAX}(\Lambda(\theta^{seg(n)}, 90, 0), \Lambda(\theta^{seg(n)}, 90, 180), \Lambda(\theta^{seg(n)}, 90, 360)) \\ \text{MNS} &= \mu_{NS}^{seg(n)} = \text{MAX}(\Lambda(\theta^{seg(n)}, 90, 135), \Lambda(\theta^{seg(n)}, 90, 315)) \\ \text{MPS} &= \mu_{PS}^{seg(n)} = \text{MAX}(\Lambda(\theta^{seg(n)}, 90, 45), \Lambda(\theta^{seg(n)}, 90, 225)) \end{aligned}$$

Where $\theta^{seg(n)}$ is the angle that the straight line between the first and the last elements of the segment form with the positive x-axis of the O-x-y plane.

Here, the function Λ represents the following function:

$$\Lambda(x; b, c) = \begin{cases} 1 - 2 \cdot \left| \left(\frac{x-c}{b} \right) \right|; & (c - \frac{b}{2}) \leq x \leq (c + \frac{b}{2}) \\ 0; & \text{Otherwise} \end{cases}$$

Another feature that can be calculated determined for a straight line is the relative length of the line with respect to the Universe of discourse. The fuzzy values associated with this characteristic are determined using three variables:

1. The horizontal length (MHLEN)
2. The vertical; length(MVLEN)
3. the slant length(MSLEN)

The following formula's can be used for the calculation of these values:

$$\begin{aligned} \text{MHLEN} &= \mu_{HLEN}^{seg(n)} = \frac{d_{P_0 P_N}^{seg(n)}}{\text{WIDTH}} \\ \text{MVLEN} &= \mu_{VLEN}^{seg(n)} = \frac{d_{P_0 P_N}^{seg(n)}}{\text{HEIGHT}} \\ \text{MSLEN} &= \mu_{SLEN}^{seg(n)} = \frac{d_{P_0 P_N}^{seg(n)}}{\text{SLANT_LENGTH}} \end{aligned}$$

Where the values WIDTH, HEIGHT and SLANT_LENGTH are depicted in **Fig. 17**, given below:

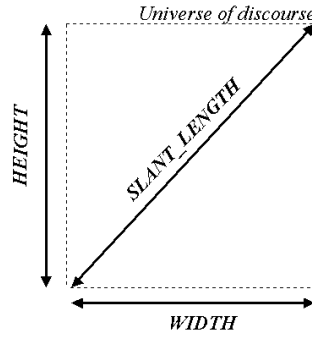


Fig. 17. Width, height and slant length of the universe of discourse

3.6.5 CALCULATION OF CURVE TYPES

If the given segment is determined to be an arc, then the following formula's can be used to calculate the fuzzy membership values in the respective sets.

3.6.5.1 C-like Curve

To distinguish a C-like curve from a D-like curve, we use the quantitative statement of the left or right convexity direction. In a curve which is vertical (i.e. either C-like or D-like), the global minimum of horizontal projections x_{\min} is relatively much lower than the weighted average of x projections, w_S, w_E , of its end points, x_S, x_E , then it is more likely to be a C-like curve.

$$MCL = \mu_{CL} = \min \left(1, \frac{\sum_{i=0}^n l_{x_i}}{n} \right); l_{x_i} = \begin{cases} 1; x_i < \frac{(x_S + x_E)}{2} \\ 0; else \end{cases}$$

In the above equation l_{x_i} is the binary function, which possesses the truth-values over the whole segment regarding the point position. i.e. segment point is on the left hand side of the median of the end point x projections, and the binary function is equal to 1, else 0. This summation function is then normalized to the universe of discourse [0,1].

3.6.5.2 D-like Curve

In a curve, which is vertical, if the global maximum of horizontal projections x_{\max} is relatively much higher than a weighted average of x projections of its end points x_S, x_E , then it is very likely to be a D-like curve. This statement is represented by the summation of a binary function r_x , which possesses truth values over the whole segment regarding the point position. i.e. if a segment point is on the right hand side

of the median of the end point x-projections, the binary function is equal to 1, else it is 0. This summation function is then normalized to the universe of discourse [0,1].

$$\text{MDL} = \mu_{DL} = \min \left(1, \frac{\sum_{i=0}^n r_{x_i}}{n} \right); r_{x_i} = \begin{cases} 1; & x_i > \frac{(x_S + x_E)}{2} \\ 0; & \text{else} \end{cases}$$

3.6.5.3 A-like Curve

In a curve which is horizontal (i.e. a A-like curve or a U-like curve), if the global maximum of horizontal projections y_{\max} is relatively much higher than a weighted average (w_S, w_E) of y projections of its end points (y_S, y_E), then it is very likely to be a A-like curve. This statement is represented by the summation of a binary function a_{x_i} , which possess' truth values over the whole segment regarding the point position. i.e. if a segment point is above the median of the end point y projections, the binary function is equal to 1, otherwise it is 0. This summation function is then normalized to the universe of discourse [0,1].

$$\text{MAL} = \mu_{AL} = \min \left(1, \frac{\sum_{i=0}^n a_{y_i}}{n} \right); a_{y_i} = \begin{cases} 1; & y_i > \frac{(y_S + y_E)}{2} \\ 0; & \text{else} \end{cases}$$

3.6.5.4 U-like Curve

In a curve which is horizontal, if the global maximum of horizontal projections y_{\min} is relatively much lower than a weighted average (w_S, w_E) of y projections of its end points (y_S, y_E), then it is very likely to be a U-like curve. This statement is represented by the summation of a binary function b_{x_i} , which possess truth values over the whole segment regarding the point position. i.e. if a segment point is below the median of the end point y projections, the binary function is equal to 1, otherwise it is 0. This summation function is then normalized to the universe of discourse [0,1].

$$MUL = \mu_{UL} = \min \left(1, \frac{\sum_{i=0}^n b_{y_i}}{n} \right); b_{y_i} = \begin{cases} 1; & y_i < \frac{(y_S + y_E)}{2} \\ 0; & \text{else} \end{cases}$$

3.6.5.5 O-like Curve

The following method was used to determine a fuzzy measure for the Olike-ness of the given segment. Let (X_{center}, Y_{center}) denotes the center of the curve. This is the same as the value calculated in determining the relative position of the curve. The expected radius of the curve is denoted by

$$rad_{EXPECTED}^{seg(n)} = \frac{(x_{max}^{seg(n)} - x_{min}^{seg(n)}) + (y_{max}^{seg(n)} - y_{min}^{seg(n)})}{4}$$

The actual radius of the curve can be determined by summing up the straight line distances from the center of the segment to each element belonging to the segment and dividing it by the number of elements in the segment

$$rad_{ACTUAL}^{seg(n)} = \frac{\sum_{k=0}^N d_{P_k(x_{CENTER}, y_{CENTER})}}{N}$$

The expected diameter for a curve with the radius is then calculated using the expression

$$diameter_{EXPECTED}^{seg(n)} = 2\pi rad_{EXPECTED}^{seg(n)}$$

The actual diameter of the given segment can be calculated by summing the straight-line distance between consecutive elements in the segment

$$diameter_{ACTUAL}^{seg(n)} = \sum_{k=0}^{N-1} d_{P_k P_{k+1}}$$

Where N stands for the number of elements in the segment.

Following equations are then carried out to determine a fuzzy value for the O-like ness of the given segment.

$$\mu_{OL2}^{seg(n)} = \begin{cases} f(x); & f(x) < 1 \\ \frac{1}{f(x)}; & f(x) > 1 \end{cases}$$

Where,

$$f(x) = \frac{\text{diameter}_{ACTUAL}^{seg(n)}}{\text{diameter}_{EXPECTED}^{seg(n)}}$$

$$\mu_{OL2}^{seg(n)} = \begin{cases} g(x); & g(x) < 1 \\ \frac{1}{g(x)}; & g(x) > 1 \end{cases}$$

Where,

$$g(x) = \frac{\text{rad}_{ACTUAL}^{seg(n)}}{\text{rad}_{EXPECTED}^{seg(n)}}$$

And,

$$\mu_{OL}^{seg(n)} = \min(\mu_{OL1}^{seg(n)}, \mu_{OL2}^{seg(n)})$$

3.7 THE DATABASE DESIGN

The entire system is centralized around a main database, which contains the information about each character, namely, the number of segments and the individual characteristics of those segments. In this work, the same database design (**Fig. 18.**) described in [8] for isolated off-line handwritten character recognition was used. The corresponding database consists of two main tables, called *Character* and *Segment*. The relationship between the two tables is a 1: m relationship, so that a character can contain many segments, but a segment can belong to one and only one character.

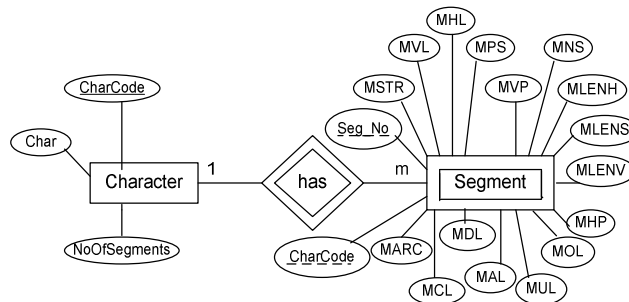


Fig. 18. The database design

3.7.1 THE TRAINING PROCESS

This is the process of building up the database. After the database is built up this can be used as an automatic rule base in the recognition process, as described in the next section. After calculating the fuzzy characteristic values for all the segments in single isolated character (after separating them to isolated characters by removing the unwanted stroked connection), those features were then recorded to the database in the following manner [8].

- 1) Calculated numerical fuzzy characteristic values of segments were then mapped to corresponding fuzzy linguistic values by using the membership function illustrated in **Fig. 19**.

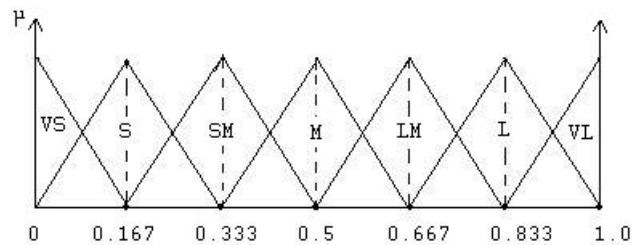


Fig. 19. Membership functions to describe the characteristics of each segments

Here, the linguistic terms VS, S, SM, M, LM, L and VL correspond to “*very small*”, “*small*”, “*small medium*”, “*medium*”, “*large medium*”, “*large*” and “*very large*”, respectively. As each fuzzy value would give a non-zero value in maximum two membership functions, the membership function with the largest membership value was selected as the corresponding linguistic value.

- 2) The system queried the user to enter the alphanumeric character corresponding to the character pattern.
- 3) A new entry was created in the table *Characters* and the corresponding values for the given character was inserted. Then an entry for each segment in the character was created in the table *Segments* and the corresponding values were recorded.

3.7.2 THE RECOGNITION PROCESS

In the recognition process the prepared database under the testing process was used as an automatic rule base as proposed in [8] for off-line isolated character recognition. In this process the characters in the document are recognized one by one. After the initial pre-processing phases namely Binarization and Skeletonization, each character skeleton was isolated as described in earlier sections. Then these each character skeleton was subjected to segmentation, calculation of fuzzy features and then recognition process independently. The recognition process for a given character skeleton was carried out as follows.

- 1) After the calculation of fuzzy features of all the segments in the given character, these features were then stored in a variable, hereafter refereed to as *ch*. This variable was a reference to an object containing information about features of all the segments of the character. For Example, the straightness of the second segment of the character could be obtained by using the variable *ch.segment[2].starightness*.
- 2) Then a query was generated and all the values of the characters stored in the database whose number of segments was equal to the given character's, were retrieved and stored in a variable, hereafter refereed to as *dbChars*. For example, the straightness of the second segment of the secondly retrieved character from the database by the previous query could be obtained using the variable *dbChars.char[2].segment[2].straightness*.
- 3) Then the min-max inference was used to get the recognized character as follows.

Then the following fuzzy relation was computed.

$$\text{Equal} = \begin{matrix} & \text{Ch} \\ \text{Db}_1 & \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \\ \text{Db}_2 & \\ \vdots & \\ \vdots & \\ \text{Db}_n & \end{matrix} \quad n \times 1$$

Where,

Ch = given character.

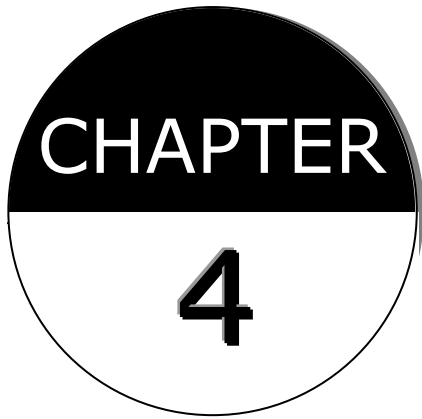
Db_i = ith character in the database.

$$x_i = \text{Min}(\text{min-max}(\text{ch.segment}[1], \text{Db}_i.\text{segment}[1]), \dots, \text{min-max}(\text{ch.segment}[m], \text{Db}_i.\text{segment}[m])). \text{ Eq}(1).$$

m = number of segments in the given character.

n = number of characters in the database having m number of segments.

Then the maximum of x_i was calculated and if that value was greater than a particular threshold value, the character in the database corresponding to that x_i value was recognized as the given character.



CHAPTER
4



4 Results & Discussions
4.1 Results & Discussions

CHAPTER 4

4.1 RESULTS AND DISCUSSION

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. Total meaningful segments from the merging process are depicted in the **Fig. 20.**

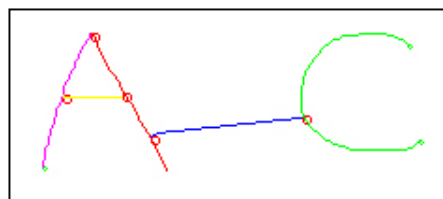


Fig. 20. Total meaningful segments from the merging process

Character : Table			
	Char_Code	Character	NoOfSegments
▶ +	9.713104292060484A	A	3
+	8.40247310567235B	B	4
+	7.788306590712521C	C	1
+	2.3262580892605005D	D	2
+	4.140543356032829E	E	5
+	13.844366043909138F	F	4
+	11.419516951109493G	G	3
+	11.558574593151096H2	H	5
+	4.813613916394347I	I	4
+	13.182534671806446J	J	2
+	5.659389969651034K	K	4
+	8.170760801283148L	L	2
+	11.50676044061547M	M	4
+	4.401243887277318N1	N	3
+	10.45111973095087O	O	1
+	4.971110107379897P	P	3
+	12.202635075407553Q	Q	2
+	13.09908137108524R	R	4
+	9.698006199152884S	S	2
+	7.29420203937546T	T	3
+	2.953505385707068U	U	1
+	6.671354440193464V	V	2
+	2.9569119217458946W	W	4
+	4.479216074072658W	W	4
+	9.002908117120121X	X	3
+	11.663254265744783Y	Y	2
+	5.9128155475357085Z	Z	3

Fig. 21. Character set with which the system was tested

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. 22). These sketches are used to formulate new input character image to an individual character recognition system [1], in order to recognize characters of the connection string.

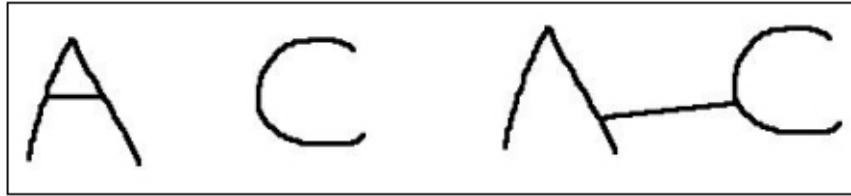


Fig. 22. Set of auto generated different *Connection sketches*

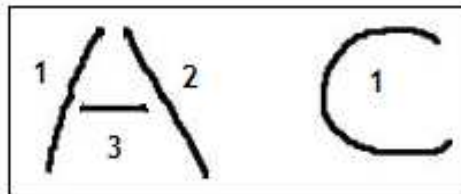


Fig. 23. Total meaningful segmentation of individual characters

Fig. 24. and Fig. 25 depict the partial database entry for character ‘A’ and ‘C’ in Fig. 23, which was used with the training character set in Fig. 21

Seg_No	MHP	MVP	MSTR	MARC	MVL	MHL	MPS	MNS	MHLEN	MVLEN	MSLEN	MCL	MDL	MAL	MUL	MOL
1	S	M	VL	VS	LM	VS	VS	SM	VS	VL	VS	VS	VS	VS	VS	VS
2	LM	M	VL	VS	M	VS	M	VS	VS	VS	VS	VS	VS	VS	VS	VS
3	M	LM	VL	VS	VS	L	VS	S	M	VS	VS	VS	VS	VS	VS	VS

Fig. 24. Partial Segment table entries for character ‘A’ in Fig. 33.

Seg_No	MHP	MVP	MSTR	MARC	MVL	MHL	MPS	MNS	MHLEN	MVLEN	MSLEN	MCL	MDL	MAL	MUL	MOL
1	M	M	M	M	VS	VS	VS	VS	VS	VS	VS	L	S	M	SM	LM

Fig. 25. Partial Segment table entries for character ‘C’ in Fig. 33

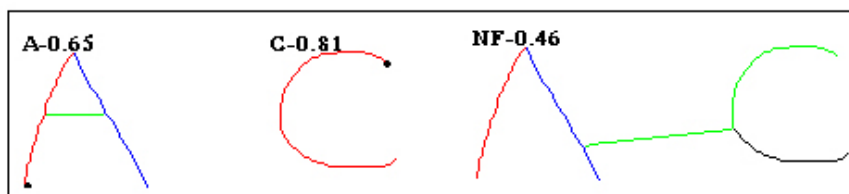


Fig. 26. Final output from the recognition system

Table. 5. The equality fuzzy values generated by the system after comparing the features of the segments of the given character (Fig. 20) with the features of segments of the characters in the database having the identical number of segments (in this case 3 segments) as the given character.

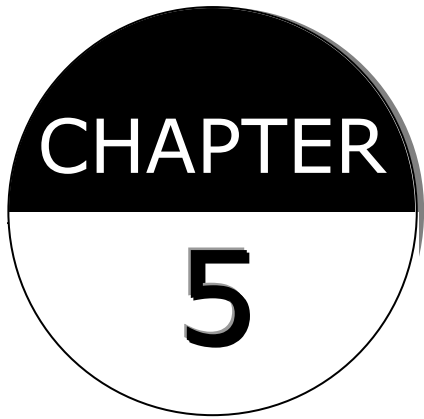
Database Character (having 3 segments)	Segment Equalities			Min
	Segment 1	Segment 2	Segment 3	
A	0.6520	0.7199	0.8257	0.6520
G	0.1579	0.4518	0.3234	0.1579
K	0.7917	0.4481	0.3635	0.3635
N	0.4286	0.7406	0.2423	0.2423
P	0.1000	0.3332	0.3226	0.1000
T	0.3333	0.5311	0.9000	0.3333
X	0.5602	0.7408	0.3336	0.3336
Z	0.3078	0.6428	0.7690	0.3078

According to **Table. 5**, it is clear that the equality between the given character pattern and the database character 'A' is 0.6520, which is the highest equality and greater than 0.5. Therefore, the system supposed to recognize the character pattern in **Fig. 20**. as character 'A'.

Table 6. The equality fuzzy values generated by the system after comparing the features of the segments of the given character 'C' (Fig. 20) with the features of segments of the characters in the database having the identical number of segments (in this case 1 segment) as the given character.

Database Character (having 1 segments)	Segment 1	Min
	C	0.8062
O	0.5405	0.5405
U	0.6571	0.6571

Likewise, the system was tested with the character patterns, which are presented in the Fig. 26. and the identified character with the final equality fuzzy values are also mentioned in the same figure.



CHAPTER
5



5 Conclusions & Recommendations

5.1 Conclusions & Recommendations

CHAPTER 5

5.1 CONCLUSIONS AND 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 stroked 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 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.