An Efficient Method for Solving Broken Characters Problem in Recognition of Vietnamese Degraded Text

Nguyen Thi Thanh Tan, Ngo Quoc Tao, Luong Chi Mai
Department of Pattern Recognition and Knowledge Engineering
Institute of Information Technology Hanoi, Vietnam
Email: {thanhtan, nqtao, lcmai}@ioit.ac.vn

***Abstract*: This paper presents an efficient method for solving the broken characters problem in the recognition of Vietnamese degraded text. Basically, the broken characters restoration process consists of three main steps: 1) analyzing and grouping connected components into connected areas; 2) building directed graph from connected areas; 3) applying a best first search A* to all its possible sub-graphs in order to an optimal strategy to rejoin the appropriate connected areas. Our experiments were carried out on the testing dataset, consists of 21690 low quality word images which are exported from 925 different quality of document pages. This method correctly finds 94.37% of the dataset.**

Keywords: Broken character, recognition, classification, restoration, bi-gram, probability, degraded text, damaged character, connected components, connected areas, directed graph.

I. INTRODUCTION

Most commercial optical character recognition systems are designed for well-formed, modern business documents. Recognizing older documents with low-quality or degraded printing is more challenging, due to the high occurrence of broken and touching characters [10], [17]. Also, in the Vietnamese Optical Character Recognition system (VnDOCR) [33], one of the most important problems that decrease the accuracy of this system is broken characters in the input document image. By definition, broken character is a character that composed of several connected components.

In the VnDOCR, the broken character problem was mainly solved in the post-processing by applying a bi-gram Vietnamese language model on the recognition results. Although they have used a technique for correcting broken characters in the character segmentation process, it is very simple. The pieces of broken characters were rejoined simply by the distance of its bounding boxes only. This approach is effective with horizontal broken characters in the simple cases, but it would fail when character is broken into a large number of components, especially with vertical broken characters as an example shown in Table 1.

Input image	*Testing result from VnDOCR*	
	Before check spelling	AFTER CHECK SPELLING
Căn cứ Nghị định số 52/1999/NĐ-CP về việc ban hành Quy chế Quản lý đầu tư và xây dựng bổ sung một số điều Bộ Tài chính sửa đổi	căn *c*~ N ~l~lị c~i~th số ~2/1999~l'Đ-CP ~ề ~ iệc ban hà~l Qu~T ~hế Quản lý đầu tư và ~ây dựlâ~ bổ .,l.ư[g một sô điều Bộ Tàl chín-h sửa đổi	căn *c*~ N ~l~lị c~i~th số ~2/1999~l'đ-cp ~ề ~ iệc ban hà~l QU~T ~hế Quản lý đầu tư và ~ây dựlâ~ bổ .,l.ư[g một sô điều Bộ Tàl chín-h sửa đổi

Table 1: Example of OCR results for broken characters

Because of the multiple generations of photocopies of the input document, the characters were broken into several pieces. Simple merging of these small pieces is not efficient because it is not clear beforehand which piece belongs to which character. Moreover, the use of n-gram language model in post-processing is usually not effective for the segmentation errors.

In this paper, we present an efficient method for correcting broken characters in recognition of Vietnamese degraded text. Our approach focuses on two main techniques: the first one for finding an optimal strategy to rejoin the appropriate connected areas into candidate characters, and the second one for classifying the damaged character image.

This paper is organized as follows: Section 2 is a review on the Vietnamese character set and their characteristic. In Section 3, we briefly address related works in dealing with broken characters. In Section 4, an efficient method to solve the broken characters in recognition of Vietnamese degraded text is proposed. Section 5 refers to the character classification method that is able to cope with damaged images. In Section 6, experimental results are analyzed in order to verify the performance of the proposed method. Finally conclusions and future developments are given in Section 7.

II. VIETNAMESE CHARACTER SET

Modern Vietnamese is written with the Latin alphabet [34], consists of 29 following letters:

- The 26 letters of the English alphabet minus f, j, w, and z.
- Seven modified letters using diacritics: đ, ă, â, ê, ô, ơ, ư

Name	***Contour***	***Diacritic***	**Accented Vowels**
Ngang	mid level	unmarked	A/a, Ă/ă, Â/â, E/e, Ê/ê, I/i, O/o, Ô/ô, Ơ/ơ, U/u, Ư/ư, Y/y
Huyền	low falling	grave accent	À/à, Ằ/ằ, Ầ/ầ, È/è, Ề/ề, Ì/ì, Ò/ò, Ồ/ồ, Ờ/ờ, Ù/ù, Ừ/ừ, Ỳ/ỳ
Sắc	high rising	acute accent	Á/á, Ắ/ắ, Ấ/ấ, É/é, Ế/ế, Í/í, Ó/ó, Ố/ố, Ớ/ớ, Ú/ú, Ứ/ứ, Ý/ý
Hỏi	dipping	hook	Ả/ả, Ẳ/ẳ, Ẩ/ẩ, Ẻ/ẻ, Ể/ể, Ỉ/ỉ, Ỏ/ỏ, Ổ/ổ, Ở/ở, Ủ/ủ, Ử/ử, Ỷ/ỷ
Ngã	glottalized rising	tilde	Ã/ã, Ẵ/ẵ, Ẫ/ẫ, Ẽ/ẽ, Ễ/ễ, Ĩ/ĩ, Õ/õ, Ỗ/ỗ, Ỡ/ỡ, Ũ/ũ, Ữ/ữ, Ỹ/ỹ
Nặng	Glottalized falling	dot below	A/ạ, Ă/ặ, Â/ậ, E/ẹ, Ê/ệ, I/ị, O/ọ, Ô/ộ, Ơ/ợ, U/ụ, Ư/ự, Y/ỵ

Table 2: Vietnamese character set

In addition, Vietnamese is a tonal language, i.e. the meaning of each word depends on the "tone" in which it is pronounced. There are six distinct tones, the first one ("level tone") is not marked, and the other five are indicated by diacritics applied to the vowel part of the syllable as shown in Table 2.

As the Thai language [22], a Vietnamese sentence consists of up to the maximum of three zones namely: the central zone (CZ), lower zone (LZ), and upper zone (CZ) as shown in Fig. 1. The central zone is limited from the baseline to mean line. This zone is the kernel of a text line. The lower zone is limited from the descender line to the baseline. For example, the dot below of Vietnamese characters in row 7th of Table 2 will belong to this zone. The upper zone is limited from the mean line to the ascender line. The diacritics, tilde, hook, acute accent, grave accent in Vietnamese language will be in this zone. Since the multi-level structure of a Vietnamese sentence, recognition of Vietnamese documents is more complicate and difficult than another language. In our approach, the zone information is obtained by using vertical histogram combined with connected components analysis algorithms.

Figure 1: Vietnamese sentence structure

III. RELATED WORKS

Many techniques have been proposed for dealing with broken characters. Basically, they can be categorized into two main approaches. The first approach is to reconstruct a complete character from a broken character, the reconstructed character not only yields more recognition accuracy, but also improves image quality [5], [6], [14], [18], [26], [28], [31]. Another approach focuses on segmentation of broken characters, recognizing them directly without reconstruction [12], [13], [16], [20], [22].

M. Droettboom [17] proposes a robust method for rejoining broken segments based on graph combinatory. The algorithm starts by building an undirected graph in which vertex represents a

connected component in the image. Two vertices are connected by an edge if the borders of the bounding boxes are within a certain threshold distance. Next, all of the different ways in which the connected component can be joined are evaluated using *k* nearest neighbor classifier. The dynamic programming is then used to find an optimal combination that maximizes the mean confidence of the characters across the entire sub-graph. However, the efficiency of this approach is based on training data. The results show that this approach segments 71% correctly when the symbol classifier has only the knowledge of complete character, and 91% when training with example from broken characters.

Basically, the proposed method in this paper is also based on graph combinatory to rejoin the appropriate connected components. However, it is different both in how the segmentation graph is built and how the optimal way is found. In addition, since our character classifier is able to cope with recognition of damaged images, the efficiency of this approach did not decrease even though it only has knowledge of complete characters.

IV. METHOD FOR BROKEN CHARACTER RESTORATION

To simplify the problem, we assume that the input of this stage is a set of low quality word image. A word is considered as a sequence of one or more characters. For the purposes of this research, these definitions will be used through this paper to describe the method in detail.

Definition 1: A *connected component* (*CC*) is a set of black pixels that are contiguous.

Definition 2: A *connected area* (*CA*) is a sequence of one or more connected components which satisfy certain given constraints.

Basically, the broken character restoration process on each input word image can be divided into three main steps:

- Connected component analysis
- Building directed graph from connected areas
- Finding an optimal solution from built graph

In the first step, all CCs from the input word images are detected and then grouped into CAs based on constraints of their bounding box. The second step will build a directed graph from these CAs. Finally, an optimal strategy to rejoin the appropriate CAs will be found at step 3 by applying a best first search A* on all possible sub-graphs.

A. Connected component analysis

As mentioned above, each complete Vietnamese character can contain maximum of three different zones as shown in the case a), b) in the Fig. 2.

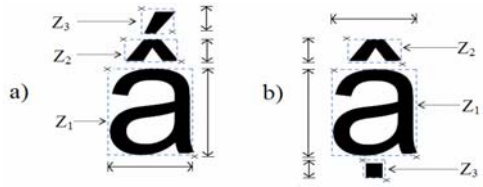


Figure 2: Multi level structure in a Vietnamese character

Symbols Z_1, Z_2, Z_3 denote three bounding boxes of zones of a character. This example show that if we consider each of CCs as a single vertex, the graph will be more complex, resulting in the significantly increase of searching time. To solve this problem, firstly we will detect all CCs from the input word image based on the edge detection algorithm [23]. Next, these CCs will be grouped together into a CA according to one of following rules.

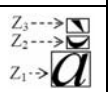
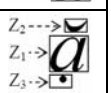
Rule 1	$Z_1 \neq \phi \wedge Z_2 \neq \phi \wedge Z_3 = \phi \wedge Z_1 \cap Z_2 = \phi \wedge$ $Top(Z_1 \cup Z_2) = Top(Z_2) \wedge$ $Bottom(Z_1 \cup Z_2) = Bottom(Z_1)$	
Rule 2	$Z_1 \neq \phi \wedge Z_2 = \phi \wedge Z_3 \neq \phi \wedge Z_1 \cap Z_3 = \phi \wedge$ $Top(Z_1 \cup Z_3) = Top(Z_3) \wedge$ $Bottom(Z_1 \cup Z_3) = Bottom(Z_1)$	
Rule 3	$Z_1 \neq \phi \wedge Z_2 = \phi \wedge Z_3 \neq \phi \wedge Z_1 \cap Z_3 = \phi \wedge$ $Top(Z_1 \cup Z_3) = Top(Z_1) \wedge$ $Bottom(Z_1 \cup Z_3) = Bottom(Z_3)$	
Rule 4	$Z_1 \neq \phi \wedge Z_2 \neq \phi \wedge Z_3 \neq \phi \wedge Z_1 \cap Z_2 \cap Z_3 = \phi \wedge$ $Top(Z_1 \cup Z_2 \cup Z_3) = Top(Z_3) \wedge$ $Bottom(Z_1 \cup Z_2 \cup Z_3) = Bottom(Z_1)$	
Rule 5	$Z_1 \neq \phi \wedge Z_2 \neq \phi \wedge Z_3 \neq \phi \wedge Z_1 \cap Z_2 \cap Z_3 = \phi \wedge$ $Top(Z_1 \cup Z_2 \cup Z_3) = Top(Z_2) \wedge$ $Bottom(Z_1 \cup Z_2 \cup Z_3) = Bottom(Z_3)$	

Table 3: Using rules in the grouping of CCs

Where *Left*(.), *Right*(.), *Top*(.), *Bottom*(.) functions is used to get the coordinates of bounding box of zones. At the end of this processing step, the coordinates of bounding box of each CA are calculated as follows:

$$Left(CA) = \min_{\forall CC \in CA} \{ Left(CC) \} \quad (1)$$

$$Top(CA) = \min_{\forall CC \in CA} \{ Top(CC) \} \quad (2)$$

$$Right(CA) = \max_{\forall CC \in CA} \{ Right(CC) \} \quad (3)$$

$$Bottom(CA) = \max_{\forall CC \in CA} \{ Bottom(CC) \} \quad (4)$$

Next step will consider all of single CCs again in order to add them into CAs if possible. Here, a CC which was bounded by the rectangle Z' is considered as a part of the CA which was bounded by the rectangle Z if they satisfy following constraints:

$$(Z \cap Z' = Z') \vee \quad (5)$$
$$(Z \cap Z' \neq Z' \wedge Left(Z \cup Z') = Left(Z) \pm \Gamma \wedge Right(Z \cup Z') = Right(Z)) \vee$$
$$(Z \cap Z' \neq Z' \wedge Left(Z \cup Z') = Left(Z) \wedge Right(Z \cup Z') = Right(Z) \pm \Gamma)$$

where Γ is a constant value, which is 0.25 times the width of Z'. Figure 3 shows results of a connected component analysis on the input word image.

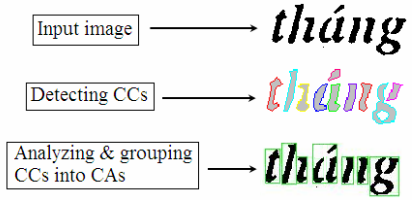


Figure. 3: Connected component analysis

B. Building directed graph

At this stage, we will build a directed graph from CAs, with each vertex represents a CA. In this graph, two vertices will be connected by an edge if the distance between its bounding box is not greater than a certain threshold.

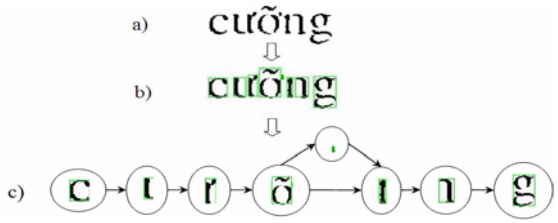


Figure 4: (a) Input image; (b) Detected CAs; (c) Building graph from CAs

This threshold is set to maximum space between two characters in a word. Since characters can be broken in the vertical and/or horizontal so the cycles can occur between CAs. Figure 4 shows one such graph.

C. Finding an optimal solution

In computer science, A* is a best-first graph search algorithm that finds the "least-cost" path from a given initial state to one goal state. It is a popular heuristic search algorithm that guarantees finding an optimal cost solution, assuming that one exists and that the heuristic used is admissible 0. Heuristic search algorithms such as A* are guided by the cost function $f(u) = g(u) + h(u)$, where $g(u)$ is the best known distance from the initial state to state u and $h(u)$ is a heuristic function estimating the cost from u to a goal state.

For the purpose of this stage, we will apply the best first search A* technique to all possible sub-graphs that were built at previous stage. Here, an optimal solution is considered as a path of the graph on which the probability of the sequence of recognized characters is highest. In order to describe the algorithm in detail, following conventional notations will be used through this section.

- u_0: is the initial state .
- *goal*: is the goal state.
- *OPEN*: is a list of states to consider to be expanded. This list is sorted by decreasing f-value.
- *CLOSE*: is a list of nodes that have been expanded. At each searching step, the best state on the open list is moved to the closed list, expanded, and its successors are added to the open list.
- $prev(u_i)$: keeps the previous state of u_i. This is a state that was selected to expand the state u_i.
- $f(u_i)$: is the cost function of the state u_i.
- $g(u_i)$: is the best known distance from the initial state to state u_i.

- $h(u_i)$: is a heuristic function estimating the cost from u_i to a goal state.
- $[ch_{u_i}, conf(ch_{u_i})]$: are the pairs of values that are obtained from character classifier (in Section 5) by classifying combinations of CAs of u_i (referred in the following), where ch_{u_i} is the recognized character, $conf(ch_{u_i})$ called confidence of ch_{u_i} is a real value in the range from 0 to 1.
- w_{u_i}: is a sequence of characters which correspond to recognized results on the path from the initial state to state u_i.
- $siblings(u_i)$: is a list of all possible states that can be expanded from u_i. This list is created by enumerating all of possible combinations of CAs from each state. To improve runtimes, the searching depth is limited by a threshold which is adjusted automatically based on the maximum number of CAs that would typically make up a single broken character and is usually less than or equal 4.

The main routing of the searching process can be described as follows;

Function Find_Optimal()
BEGIN
$OPEN = \{u_0\}$; $g(u_0) = 0$; $h(u_0)=0$; $f(u_0) = 0$;
$ch_{u_0} = NULL$; $w_{u_0} = EMPTY$; $CLOSE$ = EMPTY;
repeat
{modify the maximum estimation}
if $OPEN$ = EMPTY then
Message("There is no solution");
exit;
end if
Select u_{max} in $OPEN$ so that $f(u_{max})$ is maximum;
Pop u_{max} from $OPEN$ and push u_{max} to $CLOSE$;
Create $siblings(u_{max})$;
for each u_i in $siblings(u_{max})$ do
Call_Classifier(combination of CAs of state u_i);
$h(u_i) = \log(conf(ch_{u_i}))$;
$g(u_i) = g(u_{\max}) + \log(P(ch_{u_i} \mid w_{u_{\max}}))$;
for each u_k in $OPEN$ do
if $(u_k = u_i)$ and $g(u_k) < g(u_i)$ then
$g(u_k) = g(u_i)$;
Update the characters sequence w_{u_k} ;
Re-Calculate $f(u_k)$;
$parent(u_k) = u_{max}$;
end if
for each u_l in $CLOSE$ do
if $(u_l = u_i)$ and $g(u_l) < g(u_i)$ then
$g(u_l) = g(u_i)$;
Update the sequence of characters w_{u_k} ;
Re-Calculate $f(u_l)$;
$prev(u_l) = u_{max}$;
Propagate the change of g, w, f values to successors of u_l in $OPEN$ and $CLOSE$;
end if
if u_i is_not_exist_in($OPEN$) and u_i is_not_exist_in($CLOSE$) then
$OPEN = OPEN \cup \{u_i\}$;
$w_{u_i} = w_{u_{\max}} \cup ch_{u_i}$;
$f(u_i) = g(u_i) + h(u_i)$;
end if
end do
until $u_{max} = goal$

END

Function **Call_Classify(.)** in the algorithm is used to call the character classifier (Section 4) in order to classify the combination of CAs of state u_i, the returned value of this call is the pairs ch_{u_i} and $conf(ch_{u_i})$. Figure 5 shows an A* searching process on the sub-graphs of the example in Fig. 4, and the path with bold arrows is the optimal solution.

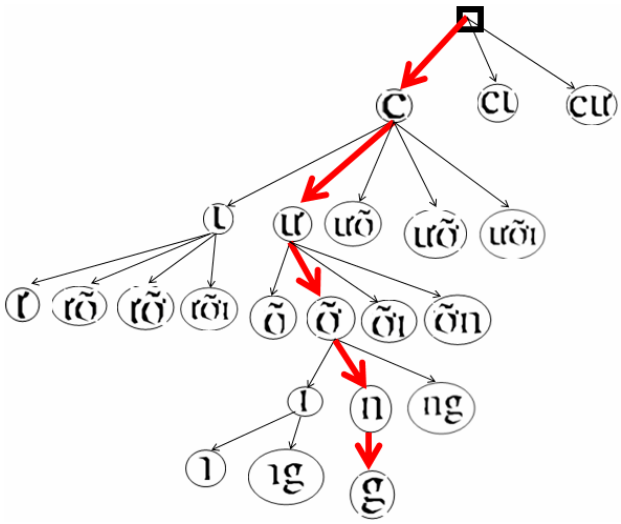


Figure 5: Applying a best first search A to sub-graphs*

In OCR, we see that reliability of a sequence of recognized results are often evaluated based on their probabilities in a given training corpus or a dictionary. For this reason, the best known *distance* $g(.)$ from the initial state to each state u_i is evaluated by the probability of a sequence of recognized characters on the path from the initial state to u_i. The heuristic function estimating the cost from u_i to a goal state is selected based on the confidence of a recognition result which is obtained by classifying combinations of CAs of the state u_i. In fact, the probability of a sequence that consist of N characters $w = ch_1ch_2\ldots ch_N$ is often calculated by applying chain rules:

$$P(w) = P(ch_1\, ch_2 \ldots ch_N) = \prod_{i=1}^{N} P(ch_i \mid ch_1 \ldots ch_{i-1}) \quad (6)$$

Since $P(ch_i \mid ch_1\ldots ch_{i\text{-}1}) \le 1 \Rightarrow P(ch_1ch_2\ldots ch_N) \le P(ch_1\, ch_2\ldots ch_{N\text{-}1})$, it means that the longer the length of a characters sequence, the smaller its probability. This can cause the high error accumulation in practice. In order to overcome this shortcoming, we use the logarithm of both probabilities and confidences instead of using them directly. The evaluation of each state in searching process can be explained more clearly as follows;

At initial state u_0: values of g, h, f are set to 0, ch_{u_0} is set to NULL, and w_{u_0} is set to EMPTY (is not any character). Assume that u_1 is one of states in *siblings*(u_0), we have:

$$h(u_1) = \log\left(conf(ch_{u_1})\right) \quad (7)$$

$$g(u_1) = \log\left(P(w_{u_1})\right) = \log\left(P(ch_{u_0}ch_{u_1})\right) = \log\left(P(ch_{u_1})\right) \quad (8)$$

Since $g(u_0)$=0 and $w_{u_0} = EMPTY$ so we can write as follow:

$$g(u_1) = g(u_0) + \log\left(P(ch_{u_1} \mid w_{u_0})\right) \quad (9)$$

In the next searching step, we assume that u_1 is the next state will be expanded, u_2 is one of states in *siblings*(u_1), we have:

$$h(u_2) = \log\left(conf(ch_{u_2})\right) \quad (10)$$

$$g(u_2) = \log\left(P(w_{u_2})\right) = \log\left(P(ch_{u_0}ch_{u_1}ch_{u_2})\right) = \log\left(P(ch_{u_1}ch_{u_2})\right) \quad (11)$$

From equation 6, we find that:

$$P(ch_{u_1}ch_{u_2}) = P(ch_{u_1}) \times P(ch_{u_2} \mid ch_{u_1}) \quad (12)$$

Therefore:

$$\begin{aligned} g(u_2) &= \log\left(P(ch_{u_1}) \times P(ch_{u_2} \mid ch_{u_1})\right) \\ &= \log\left(P(ch_{u_1})\right) + \log\left(P(ch_{u_2} \mid ch_{u_1})\right) \\ &= g(u_1) + \log\left(P(ch_{u_2} \mid w_{u_1})\right) \end{aligned} \quad (13)$$

In general, for the state u_k we will have:

$$h(u_k) = \log\left(conf(ch_{u_k})\right) \quad (14)$$

$$g(u_k) = g(parent(u_k)) + \log(P(ch_{u_k} \mid w_{prev(u_k)})) \quad (15)$$

If we assume that $w_{prev(u_k)} = ch_0ch_1\ldots ch_n$ is the sequence of recognized characters on the path from initial state to state $prev(u_k)$, the posterior probabilities $P(ch_{u_k} \mid w_{prev(u_k)})$ will be calculated as follows:

$$P(ch_{u_k} \mid w_{prev(u_k)}) = \begin{cases} P(ch_{u_k}) & if\ w_{prev(u_k)} = EMPTY \\ P(ch_{u_k} \mid ch_0ch_1..ch_n) & otherwise \end{cases} \quad (16)$$

Up to now, applying the maximum likelihood estimation method (MLE), we have:

$$P(ch_{u_k}) = \frac{freq(ch_{u_k})}{N} \quad (17)$$

where N is the total number of characters in the training corpus.

$$P(ch_{u_k} \mid ch_0ch_1\ldots ch_n) = \begin{cases} \dfrac{freq\,(ch_0ch_1\ldots ch_nch_{u_k})}{freq\,(ch_0ch_1\ldots ch_n)} & if\ \ freq\,(ch_0ch_1\ldots ch_n) \neq 0 \\ 0 & otherwise \end{cases} \quad (18)$$

where *freq*(.) denotes the number of occurrences of a sequence of characters in the training corpus.

The introduction of the context information in searching process will help it to avoid the paths that have not the correct result. In our approach, the statistical information are evaluated based on a training corpus consists of 7178 single words from Vietnamese word dictionary. The longest Vietnamese word consists of 7 characters, for example in the case of the word “nghiêng”. In order to improve runtimes,

all of character strings and its posterior probabilities are stored in the form of a tree structure called MixTree. Basically, this is a mixture of the binary tree search and the Trie data structure 0, in which each node consists of five data fields as follows:

- **Key**: is a character
- **Info**: keeps the information of current node including the posterior probability of a character string that is terminated by its key.
- **Child**: point to its child node.
- **Left**: point to its left sibling node.
- **Right**: point to its left sibling node.

Each node must be greater than its left node and smaller than its right node. It means that for this structure each node in company with its left node and its right node are organized in the form of a binary tree search while the sequences characters themselves are represented implicitly as paths to a node. For example, these character string "bố", "bế", "bống", "ai", "an", "anh", "cô", "cành", "của", "ông" will be represented by the MixTree as in Fig. 6.

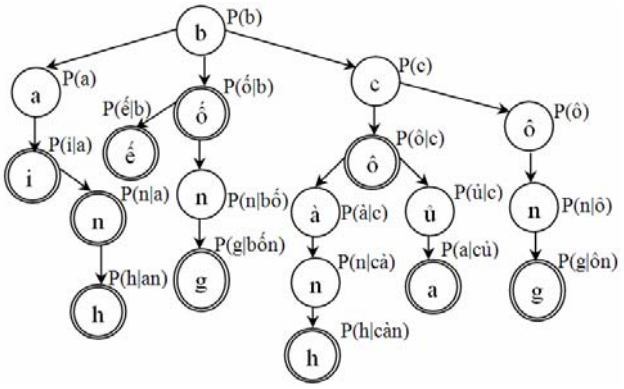


Figure 6: Data structure for storing a lexical of character strings

Although this data structure does not save space as much as the Dawg structure [35], it has advantages in searching speed because it takes advantage of the binary tree search.

V. CHARACTER CLASSIFIER

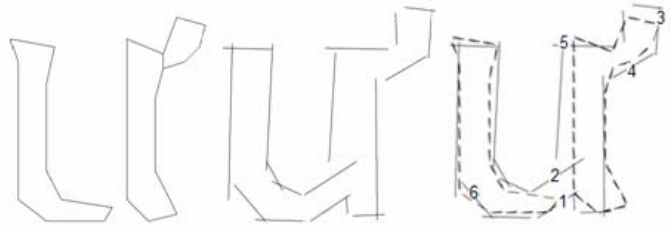
The accuracy of almost all OCR systems depends directly on the character classification process. Currently, many character classification methods are proposed, including template matching methods 0, 0, 0, statistical classification methods such as the naive Bayesian classifier [3], [27], k-nearest neighbor (K-NN) [4], [29], [32], artificial neural networks (ANNs) [2], [7], [19], support vector machines (SVMs)[1], 0, and hidden Markov models (HMMs) [1], [11], [24]. Most of these methods gain the high accuracy on high quality images. But in the case of damaged images including broken or touching characters, accuracy of these method are guaranteed only if they have known about almost types of damaged images, i.e classification algorithms must be trained with almost different types of damaged images. It means that in order to apply these methods effectively, we must have a great and complete training database. This takes us a lot of time and effort. In order to overcome this shortcoming, we use the breakthrough solution 0, 0 for features extraction in our character classification model. This is the idea that allows the features in the input image need not be the same as the features in the training data.

During training, the segments of a polygonal approximation are used for features called prototype features. All of these features then will be clustered into *template*s. For the purpose of this approach, a template will consists of clustered prototype features which are representatives of a character class.
In the classification, features of a small, fixed length (in normalized units) are extracted from the outline and matched many-to-one against the clustered prototype features of templates. Owing to the process of small features matching large prototypes, this algorithm is easily able to cope with recognition of damaged images.

In fact, to improve runtimes, each template is represented by a logical sum-of-product expression with each term called a *configuration*. Each feature extracted from input image looks up a bit vector of *template*s of the given class that it might match, and then the actual similarity between them is computed (this value was clearly defined in [15]). The matching process keeps a record of the total similarity evidence

of each feature in each configuration, as well as of each template. Here, the result of each character classification process is represented by two values: recognized character denoted *ch* and its confidence denoted *conf*(*ch*). The confidence is considered as the best combined distance, which is calculated from the summed feature and prototype evidences and the recognized character is the label of character class having the best combined distance.

The features extracted from the input image are thus 3 dimensional, (x, y position, angle), with typically 50-155 features in a character, and the prototype features are 4-dimensional (x, y, position, angle, length), with typically 10-40 features in a template configuration.



(a) (b) (c)

Figure 7: (a) Input image; (b) Prototype; (c) Matching result

For example in Fig. 7c, the short, thick lines are the features extracted from the input image, and the thin, longer lines are the clustered segments of the polygonal approximation that are used as prototypes. Features labeled 1, 2, 3 are completely unmatched, features labeled 4, 5, 6 are unmatched, but, apart from those, every prototype and every feature is well matched.

VI. EXPERIMENTS AND RESULT

The success of the proposed method is affected directly by the accuracy of the character classification algorithm and the broken character restoration process. For the purpose of this research, our experiment will focus on two main processes:

- Experimenting on the character classification process.
- Experimenting on the broken character restoration process.

The first process is performed to evaluate the accuracy of the character classification method for the various qualities of input images, especially with damaged images. In the second process, the experimental results are analyzed in order to verify the performance of the proposed method.

A. Experimenting on the character classification

1) Training data

For the Vietnamese character classification, we used a training data with 185 characters classes, consist of:

- The digits from 0 to 9.
- The upper/lower letters of English alphabet from A to Z, a to z.
- The upper/lower of Vietnamese alphabet with its tone: à ả ã á ạ ă ằ ẳ ẵ ắ ặ â ầ ẩ ẫ ấ ậ đ è ẻ ẽ é ẹ ê ề ể ễ ế ệ ì ỉ ĩ í ị ò ỏ õ ó ọ ô ồ ổ ỗ ố ộ ơ ờ ở ỡ ớ ợ ù ủ ũ ú ụ ư ừ ử ữ ứ ự À Ả Ã Á Ạ Ă Ằ Ẳ Ẵ Ắ Ặ Â Ầ Ẩ Ẫ Ấ Ậ Đ È Ẻ Ẽ É Ẹ Ì Ỉ Ĩ Í Ị Ò Ỏ Õ Ó Ọ Ô Ồ Ổ Ỗ Ố Ộ Ơ Ờ Ở Ỡ Ớ Ợ Ù Ủ Ũ Ú Ụ Ư Ừ Ử Ữ Ứ Ự Ỳ Ỷ Ỹ Ý Ỵ.

In the training data, each character class will be trained with a mere 30 samples of 185 characters from 6 typical Vietnamese fonts (.VnTime, Times New Roman, Arial, Tahoma, Courier New, Verdana) in a single size, but with 4 attributes (regular, bold, italic, bold italic), making a total of 133200 training samples.

2) Testing results

Testing data	Number of characters	Complete characters	Broken characters	Touching characters
DataSet 1	40500	97%	2%	1%
DataSet 2	55700	85%	12%	3%
DataSet3	45200	62%	32%	6%

Table 4: Distribution of character types of the input data

In order to evaluate the efficiency of this method in recognition of optical Vietnamese characters, we used three data sets collected from the books, magazines and documents in difference qualities. The

distribution of the types of characters in the theses data is given in Table 4.

The character classification algorithm is not only experimented on these data, but also compared with the accuracy of character classifier of VnDOCR 3.0 system [33]. Experiment results are shown in Fig. 8.

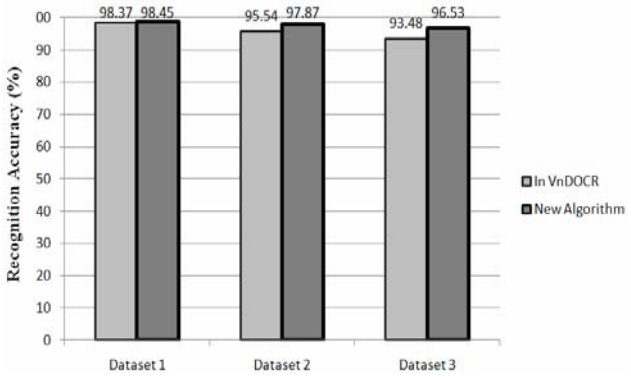


Figure 8: Accuracy of character classification algorithm

From these experiments, we find that: with the testing data dataset 1 consisting almost of high quality images, the accuracy of both algorithms is equivalent (gaining over 98%). However, in case of the number of broken and touching characters increases in the dataset 2 and dataset 3, the accuracy of this algorithm is 2% to 3% higher than classification algorithm of VnDOCR 3.0 .

B. Experimenting on the broken character restoration

1) Experimental data

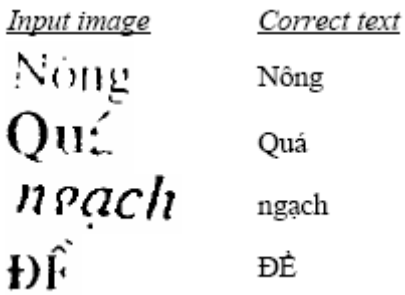
Our experiments were carried out on 925 page images scanned at 300 dpi. These images are a mixture of real office documents varying in quality from original business letters, book and magazine pages to badly degraded photocopies and faxes.

2) Experimental results

The experimental process begins by using VnDOCR to recognize all input document page images. In this step, all of words which could not be correctly recognized by this system will be exported to a dataset called the low quality word images dataset. Here, we have extracted total of 21690 low quality word images from 925 input pages, some of them are displayed in the Fig. 9. This dataset will be used to evaluate the efficiency of the proposed method. We find that almost words in this exported dataset are broken into multi fragments both vertically and horizontally.

Figure 9: Dataset of Low quality Vietnamese word images

Our experiments were carried out on PC Intel® Pentium® Dual Core Processor 2.4 GHz, 1 GB of RAM, Window XP operating system. The experiment shows that this process finds 20469 words exactly, corresponding to 94.37% of the input data set. From these recognized results, we find that almost all cases of errors are caused by the failure of the character classification when input images looses important components (features) or are distorted greatly such as following examples.



Apart from those, our method performs very well on the dataset, not only for simple cases of broken characters, but also for the complex cases in which characters were broken into multi fragments both vertically and horizontally. The time to process on the input word image with average size of 144×64 pixels and consists about 8 connected components is relatively estimated about 0.036s.

3) Limitation of the method

Although the proposed method is able to deal with broken characters in recognition of low quality documents images, it consumes more computation time than the early method of us in VnDOCR system in the case of high quality documents images. Therefore, in our system, this method will be applied only for the words of input documents, images that were not recognized well enough from the previous stage.

VII. CONCLUSIONS

In this paper, a method to deal with the broken characters problem in recognition of Vietnamese degraded text is proposed. This method performs very well on the experiment data. It is easily able to cope with recognition of broken characters even if they are split into a large number of connected components. From the experimental results, we can conclude that the proposed method will be useful in significantly improving the recognition rate of Vietnamese character recognition systems.

ACKNOWLEDGMENT

In specially, we would like to thank NAFOSTED project NCCB 2009 for funding support us to fulfill this paper. We also would like to thank the Department of Pattern Recognition and Knowledge Engineering of the Institute of Information Technology for encouraging research the question, designing or conducting the experiments.

REFERENCES

[1] A. R. Ahmad, C. Viard-Gaudin, M. Khalid, *"Lexicon-Based Word Recognition Using Support Vector Machine and Hidden Markov Model"*, ICDAR09, pg 161-165, 2009.

[2] A. Rehman, D. Mohamad and G. Sulong, *"Implicit Vs Explicit based Script Segmentation and Recognition:A Performance Comparison on Benchmark Database"*, Int. J. Open Problems Compt. Math., Vol. 2, No. 3, pg 252-263, 2009.

[3] A. Barta, I. Vajk, *"Integrating Low and High Level Object Recognition Steps by Probabilistic Networks"*, International Journal of Information Technology, 2006.

[4] I. Adnan, A. Rabea, S. Alkoffash Mamud and M. J. Bawaneh, *"Arabic Text Classification using K-NN and Naive Bayes"*, Journal of Computer Science 4 (7), pg 600-605, 2008.

[5] B. Gatos and K. Ntirogiannis, *"Restoration of Arbitrarily Warped Document Images Based on Text Line and Word Detection"*, SPPRA (2007), pp. 203-208, 2007.

[6] B. Gatos, I. Pratikakis, and K. Ntirogiannis, *"Segmentation based recovery of arbitrarily warped document images"*. Proc. Int. Conf. Document Analysis and Recognition, 2007.

[7] C. L. Liu and H. Fujisawa, *"Classification and learning methods for character recognition :Advances and remaining problems"*, in Machine Learning in Document Analysis and Recognition, pp. 139.161, 2008.

[8] C-N. E. Anagnostopoulos, *"License Plate Recognition From Still Images and Video Sequences: A Survey"*, IEEE Transactions On Intelligent Transportation Systems, Volume 9, pg 378, 2008

[9] O. Golubitsky, S. M. Watt, *"Online Computation of Similarity between Handwritten Characters*", Proc. Docum. Rec and Retrieval (DRR XVI) , 2009, C1–C10.

[10] H. Fujisawa, "*A View on the Past and Future of Character and Document Recognition"*, ICDAR07, pg. 3-7, vol 1, pp. 3-7, 2007.

[11] A. Al-Muhtaseb, S. A. Mahmoud, R. Qahwaji, *"Recognition of off-line printed Arabic text using Hidden Markov Models*", Signal Processing 88(12), pg 2902-2912, 2008.

[12] N.R. Howe, F. Nicholas, S.L.Shao-Lei, R. Manmatha, *"Finding words in alphabet soup: Inference on freeform character recognition for historical scripts"*, PR(42), No. 12, December 2009, pp. 3338-3347,.

[13] C. Jacobs, P.Y. Simard, P. Viola, J. Rinker, *"Text recognition of low-resolution document images"*, ICDAR05, pg 695-699. 2005.

[14] J. V. Beusekom, F. Shafait, T. M. Breuel, *"Image-Matching for Revision Detection in Printed Historical Documents",* 29th Annual Symposium of the German

Association for Pattern Recognition, DAGM'07, Heidelberg, Germany. Sep.,2007.

[15] D. S. Johnson, D. M. Seaman, *"Noise tolerant optical character recognition system"*, United States Patent 5237627

[16] V. Lavrenko, Rath, T.M.[Toni M.], Manmatha, R., *"Holistic word recognition for handwritten historical documents",* DIAL04, pg 278-287, 2004.

[17] M. Droettboom, *"Correcting broken characters in the recognition of historical printed documents,"* In Proceedings of the third ACM/IEEE-CS joint conference on Digital libraries, pp. 364-366, 2003.

[18] M. Gevrekci, B. K. Gunturk, Y. Altunbasak, *"Restoration of Bayer-sampled Image Sequences"*, Comput. J. 52(1), pg 1-14, 2009

[19] N. F. Shilbayeh, M. Z. Iskandarani, *"Effect of Hidden Layer Neurons on the Classification of Optical Character Recognition Typed Arabic Numerals",* Journal of Computer Science 4, pg 578-584, 2008.

[20] N. Doulgeri, E. Kavallieratou: "*Retrieval of historical documents by word spotting"*, DRR09, pg 1-10, 2009.

[21] P. Hart, Nils Nilsson, and Bertram Raphael, *"A Formal Basis for the Heuristic Determination of Minimum-Cost Paths"*, IEEE Transactions of Systems Science and Cybernetics, SSC-4(2):100–107, 1968.

[22] P. W. Yingsaeree and A. Kawtrakul, *"The Utilization of Closing Algorithm and Heuristic Information for Broken Character Segmentation"*, IEEE conference on Cybernatics and Intelligent Systems (CIS2004), Singapore, 2004.

[23] P.J. Rousseeuw, A.M. Leroy, *"Robust Regression and Outlier Detection"*, Wiley-IEEE, 2003.

[24] P. Natarajan, K. Subramanian, A. Bhardwaj, R. Prasad, *"Stochastic Segment Modeling for Offline Handwriting Recognition"*, ICDAR09, pg 971-975, 2009.

[25] R. Smith, *"An Overview of the Tesseract OCR Engine"*, ICDAR 2007, Vol 2, pp.629-633, 2007.

[26] S. Lu and C. L. Tan. *"The restoration of camera documents through image segmentation"*. In 7th IAPR workshop on Document Analysis Systems, pg 484–495, 2006.

[27] J. Sung, Bang S.J., Choi S., *"A Bayesian network classifier and hierarchical Gabor features for Handwritten Numeral Recognition"*, Pattern Recognition Letters, pg 66-75, 2006.

[28] Tan, C.L.Chew Lim, Zhang, L.Li, Zhang, Z.Zheng, Xia, T.Tao, *"Restoring Warped Document Images through 3D Shape Modeling"*, PAMI(28), No. 2, February 2006, pp. 195-208.

[29] Y. Zhou, Y. Li, S. Xia, *"An Improved KNN Text Classification Algorithm Based on Clustering*". JCP 4(3), pg 230-237, 2009.

[30] Y-Y. Chiang, and C. A. Knoblock, *"Classification of Line and Character Pixels on Raster Maps Using Discrete Cosine Transformation Coefficients and Support Vector Machines"*, Proc. Int. Conf. Pattern Recognition (ICPR'06), 2006.

[31] Y. Zhang, C. Liu, X. Ding, Y. Zou, *"Arbitrary warped document image restoration based on segmentation and Thin-Plate Splines"*, ICPR 2008, pg 1-4

[32] Z. Voulgaris, G. D. Magoulas, *"Extensions of the k nearest neighbour methods for classification problems"*, Proc. the 26th IASTED Conference on Artificial Intelligence and Applications, Innsbruck, Austria, Feb. 2008, pp. 23-28.

[33] http://www.vndocr.com/

[34] http://www.cjvlang.com/Writing/writviet.html

[35] http://en.wikipedia.org/wiki/Directed_acyclic_word_graph

[36] http://en.wikipedia.org/wiki/Trie

AUTHORS' BIOGRAPHIES

Nguyen Thi Thanh Tan worked at the Institute of Information Technology (IOIT), Vietnamese Academy of Science and Technology (VAST) in 1983. She received the B. S. degree in 1999. She received Msc degree in 2004. From 2003 to now, she is the PhD student at Institute of Information Technology. Her research interests include image processing, optical character recognition, document retrieval and statistical approaches in machine learning.

Ngo Quoc Tao worked at the Institute of Information Technology (IOIT), Vietnamese Academy of Science and Technology (VAST) in 1983. He received a B.Tech. degree in mathematics from Hanoi University of Technology (1982), Ph.D. degrees in

Image Processing (1997) and Associate professor (2002) from VAST/IOIT.

His main research interests are image processing and pattern recognition including image retrieval, automatic data entry, test for schools and universities, image features extracting, image vectorization and map generalization.

Luong Chi Mai received the B. S. degree in applied mathematics from Kishinov University, USSR (former), in 1981. Then she joined to the Institute of Informatics, Hanoi as a junior researcher. She received PhD degree in computer science and Associate Professor in 1991 and 2005 respectively. Now she is a principal researcher and a Head of the Department of Pattern Recognition and Knowledge Engineering, Institute of Information Technology in Hanoi. Her research interests include speech recognition and synthesis, statistical approaches in machine learning, human machine interaction. Email: lcmai@ioit.ac.vn