A Lexicon Driven Approach for Off-line Recognition of Unconstrained Handwritten Korean Words

Soo H. Kim*  S. Jeong†  Ching Y. Suen‡
*Department of Computer Science, Chonnam National University
300 Yongbond-dong, Buk-gu, Kwangju 500-757, Korea
{shkim, swjong}@chonnam.chonnam.ac.kr
†Center for Pattern Recognition and Machine Intelligence
Concordia University, Suite GM606, 1455 de Maisonneuve Blvd. West
Montreal, Quebec H3G 1M8, Canada
suen@cenparmi.concordia.ca

Abstract

We propose a new method for the recognition of unconstrained handwritten words consisting of Korean and numeric characters. To overcome the difficulty in separating touching characters, we adopt an over-segmentation technique and we find the optimal segment combination using a lexicon-driven word scoring technique and a nearest neighbor classifier. The optimal combination gives the final segment positions for individual characters with the best matching word in the lexicon. The proposed system has yielded an accuracy of 90.64% for 908 word images on live mail pieces.

1. Introduction

Off-line recognition of unconstrained handwritten words is one of the hottest subjects in pattern recognition. This technology has a number of applications such as reading addresses on mail pieces, reading legal amounts on bank checks, processing of tax forms, routing FAX messages, and so on.

Hundreds of approaches have been reported so far for Roman-style cursive script recognition. Among them, Hidden Markov Model (HMM)-based approaches [1,2,3], and segmentation-based Dynamic Programming (DP) techniques [4,5,6,7] have emerged with general popularity. Both of them use the same over-segmentation strategy, in which the input word image is first sliced into small segments containing a part or possibly all of one character image, to resolve the problem of separating touching characters.

The HMM-based approaches are not adequate for the recognition of Korean words, because the model representing a set of Korean words and/or characters becomes huge due to thousands of states and observation symbols. It means millions of HMM parameters have to be estimated, and hence results in a need of impractically large amount of training data. Similarly, the DP techniques are not efficient in Korean word recognition, because of its time complexity, which is proportional to the number $C$ of character classes at hand. In case of English words, the value of $C$ is less than 52, but it could be more than 2,000 in Korean words.

Most existing research on Korean word recognition aims at reading addresses [8,9,10,11], since it is possible to construct a high accuracy word recognition system by using post-processing with a lexicon. However, they all assume the complete separation for input word images and it prevents the methods from being used in practice.

In this paper, we propose a method for recognizing unconstrained handwritten words consisting of Korean and numeric characters. One of our major contributions is to lighten the problem of computational burden that occurs when a conventional segment-then-recognizer technique is applied to Korean word recognition. We have built some speed-up strategies based on the unique characteristics of Korean words and characters; Korean characters have almost the same width, Korean words are composed of a few characters, and the number of character classes appearing in a specific position of an word in a lexicon is far smaller than that of entire classes in the lexicon. Besides, we have avoided unnecessary computations by introducing the lexicon as early as possible. All these speed-up strategies make our system suitable for real-time applications.

2. Proposed system

In this section, we describe the proposed handwritten word recognition system by each module. The system
depicted in Figure 1 takes a segment-then-recognize approach as basic paradigm and introduces the concept of over-segmentation. The system yields an ordered list of words in the lexicon according to the matching scores between the sliced segments and all lexicon words.

![System structure](image)

**Figure 1. System structure**

### 2.1 Vertical Slicing

In this module an input word image containing \( N \) characters is split vertically into \( M \) segments, \( S_1, S_2, \ldots, S_M \) (\( M \geq N \)). It is important to possibly split the word image at borders between characters for the next module to group one or more segments into well-defined character images. We impose the following two assumptions on segments \( S_m, m=1, \ldots, M \).

1. Any segment \( S_m \) represents a partial or full character.
2. The number of segments into which a character may be split can not exceed a fixed threshold \( \alpha \).

The first assumption indicates that all touching between characters should be separated and the second is to avoid generating extraordinary segment groups in the next module. Figure 2 shows the result of vertical slicing, where an image containing 3 characters is split into nine segments \( S_1, S_2, \ldots, S_9 \).

![Vertical slicing](image)

**Figure 2. Example of vertical slicing: (a) input word image, (b) sliced image**

### 2.2 Combination Generation

Let a *group* be a set of one or more consecutive segments and a *combination* be a set of consecutive groups. A group will be regarded as representing a character, while a combination of \( N \) groups will be matched with the words in a lexicon \( L_N \) consisting of words of length \( N \). The combination generation module finds all possible combinations for words of all possible lengths.

The values \( M \) and \( N \) should be known before the combination generation. The value \( M \) is known by the vertical slicing module, but the maximum and minimum numbers of \( N \) should be extracted from the lexicons. Since Korean words are composed of quite a few characters - two to five in general, we assume that \( N_{\min} \) is 2 and \( N_{\max} \) is 5 from now on.

A combination made up of \( N \) groups can be expressed as a function that indicates at which \( N \) segment groups end. The function can be described as follows:

\[
\begin{align*}
\text{fs} : \{0, 1, 2, \ldots, N\} &\to \{0, 1, 2, \ldots, M\}, \\
\text{where (1) } & N \leq M \\
(2) f_s(0) &= 0, \quad f_s(N) = M, \quad \text{and} \\
(3) f_s(n) - f_s(n-1) &\leq \alpha, \quad \forall n \in \{1, 2, \ldots, N\}.
\end{align*}
\]

0 in both sets is a null (null group or null segment) and the other elements indicate indices of groups and segments for the input image respectively. The condition (3) comes from the second assumption imposed for a segment. The \( n \)-th group \( G_n \) in a combination of size \( N \) can be represented as

When we define a set of all possible combinations \( G_{N} = S_{f_s(n-1)+1} \ldots S_{f_s(n)}, 1 \leq n \leq N \) consisting of \( N \) groups \( G_1 \ldots G_N \) as \( F_{\text{MM}} = \{ f_s \} \), the goal of this module detects all possible combinations \( F_{\text{MM}} \) \( N_{\min} \leq N \leq N_{\max} \). A dynamic programming method is used as follows to find the combinations in \( O(NM) \) time.

**Step1. Initialization**

\[
F_{\text{MM}} \equiv \{ f_s : \{0,1\} \to \{0,1\} \}, \quad f_s(0) = 0, \ f_s(1) = 1.
\]

**Step2. Recursion for** \( N_{\min} \leq N \leq N_{\max} \) and \( N \leq m \leq M \)

\[
F_{\text{MM}} = F_{\text{MM}} \otimes m \cup F_{N, f_s-1} \otimes \emptyset.
\]

**Step3. Output**

\[
F_{N_{\min}} \leq N \leq N_{\max}.
\]

where \( F_{N,f_s-1} \otimes m \) means modifying every \( f_s(N) \) in \( F_{N,f_s-1} \) from \( m-1 \) into \( m \), and \( F_{N,f_s-1} \otimes \emptyset \) \( m \) means equating \( f_s(i) \) in \( F_{N,f_s-1} \) to \( f_s(N) \), \( 1 \leq i \leq N-1 \), and adding \( f_s(N) = m \) at the end of every combination.

The number of combinations is closely related to the speed of the proposed system since it finds the best match by calculating matching scores between every word in the lexicon and every combination generated from the input image. The number can be reduced remarkably by considering \( \alpha \), the maximum number of segments that a character can be split.

### 2.3 Group Filtering

There exist many groups whose constituent segments are the same in the combinations. This may result in possibly redundant feature extraction and character
recognition. The objective of group filtering module is to avoid these redundant calculations by finding out a set of non-redundant groups among all the combinations.

Let’s define the non-redundant groups as $G(k)$, $1 \leq k \leq \sum_{i} N_{G(i)}$ where $N_{G(i)}$ is the number of the non-redundant groups. We find the set of non-redundant groups by a simple linear search for the combinations. If a non-redundant group $G(k)$ is found, information about the position of the group is maintained. This is because we would like to match $G(k)$ with only the character classes appearing in the same position of the lexicon words. In this regard we define a unigram $l(j)$ as a set of character classes appearing in the $j$-th character position of all words of length $i$. We store such position information in $\text{MASK}(G(k)) = \{m_{ij}\}$. If $m_{ij} = 1$, $G(k)$ appears in the $j$-th position of the combination consisting of $i$ groups.

### 2.4 Feature Extraction

After group filtering, features from non-redundant group images are extracted using the method of Yamashita [12]. We have selected a $9 \times 7$ mesh based on the statistics of aspect ratio of height versus width and extracted four directional segment features in each cell of the mesh. The four directions are horizontal, vertical, left-diagonal, and right-diagonal. Consequently, each non-redundant group image is transformed into a $252(4 \times 9 \times 7)$ dimensional feature vector.

### 2.5 Character Recognition

Next, we compute class-matching scores for each non-redundant groups. Character classes considered for a group are determined by its $\text{MASK}$. In other words, we can get the character classes by a union of the classes in the unigrams that the $\text{MASK}$ indicates. The matching score $d(C, G(k))$ between a group $G(k)$ and a character class C has been implemented with a modified $k$-NN ($k$-nearest neighbor) classifier. It finds $k$ prototypes which are nearest to $G(k)$ among prototypes of the class C and computes $d(C, G(k))$ as an average of the $k$ distances.

### 2.6 Word Scoring

The final goal of our word recognition system is to find the best matching word for an input image in the lexicons $\text{L}_{x}$, $N_{\text{min}} \leq N \leq N_{\text{max}}$. The system has already split the input word vertically into $M$ segments and then generated all possible combinations via $\text{F}_{\text{opt}}$, $N_{\text{min}} \leq N \leq N_{\text{max}}$. Given a word $W$ in a lexicon $\text{L}_{x}$, the following function gives the best matching score of $W$ with respect to all combinations in $\text{F}_{\text{opt}}$.

$$D_{N}(W) = \min_{f \in \text{F}_{\text{opt}}} \left\{ \frac{1}{N} \sum_{n=1}^{N} d(C_n, S_{f(n-1)+1} \cdots S_{f(n)}) \right\}$$

where $W = \{C_1, \ldots, C_N\}$, and $d(\cdot)$ represents the distances from class $C_n$ to $G_n = S_{f(n-1)+1} \cdots S_{f(n)}$. The best matching word for the input image is determined as the word with the minimum matching score $D$ defined as

$$D = \min_{N_{\text{min}} \leq N \leq N_{\text{max}}} D_{N} = \min_{W \in \text{F}_{\text{opt}}} D_{N}(W),$$

where

### 3. Experimental results

#### 3.1 Environment

The proposed system has been implemented using a C programming language on a Pentium 200MHz PC and tested with 908 word images collected from live mail pieces in Korea which were scanned in a resolution of 200 DPI(dots per inch). The word images were extracted and labeled manually. The number of constituent characters within a word image varies from 2 to 5 as summarized in Table 1. Figure 3 shows some examples of the test images.

| Table 1. The number of test word images
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word length</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Total</td>
</tr>
<tr>
<td>No. test images</td>
<td>148</td>
<td>507</td>
<td>164</td>
<td>89</td>
<td>908</td>
</tr>
</tbody>
</table>

![Figure 3. Examples of test word images](image)

Our system can be trained easily because only the prototypes used by the character recognition module have to be prepared. Prototypes for Korean characters have been trained with a handwritten Korean character database, namely PE92 [13], while those for numerals have been trained with handwritten numeral samples collected privately.

#### 3.2 Performance

Table 2 summarizes the accuracy of our handwritten word recognition system (HWRS in short, hereafter) when tested with the 908 images. As can be seen from the table, the recognition accuracy differs according to word length.

| Table 2. Word Recognition accuracy
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Word length</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>Total</td>
</tr>
<tr>
<td>Accuracy(%)</td>
<td>92.6</td>
<td>92.0</td>
<td>92.1</td>
<td>77.5</td>
<td>90.64</td>
</tr>
</tbody>
</table>

Table 3 shows some statistics obtained during the processing of word images. The first column of the table shows the number $M$ of segments generated by the vertical slicing module. The value $M$ varies from 4 to 24 depending on the manner of touching between characters.
as well as the image width. The average value of $M$ over all the input images is 10.95.

The third column shows the number of combinations given a set of segments. The number of combinations for grouping $M$ consecutive segments into $N$ consecutive groups is $\binom{M}{N}$, but we can reduce it by considering $\alpha$ in the combination generation module.

The next column shows the number of non-redundant groups in the entire combinations. Only these groups are considered for recognition. One can see that on the average only 44.27 groups remain after the group filtering module.

The last column shows the average size (number of character classes) of the unigram associated with each non-redundant group. A 252-dimensional feature vector computed from each non-redundant group would be compared to the prototypes for this number of classes. By matching a group with the classes in the unigram, instead of all the 362 classes at hand, we can reduce the computation cost of the character recognition module.

In summary, 44.27 times of feature extraction and character classifications with respect to 61 classes are performed on average to recognize an input word, and the average processing time is 2.83 seconds.

<table>
<thead>
<tr>
<th>No. Segments</th>
<th>No. Test image</th>
<th>No. Combinations (with $\alpha$)</th>
<th>No. Non-redundant Groups</th>
<th>Average unigram size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>58.4</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>15</td>
<td>15</td>
<td>68.1</td>
</tr>
<tr>
<td>6</td>
<td>51</td>
<td>30</td>
<td>20</td>
<td>78.2</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>54</td>
<td>25</td>
<td>85.6</td>
</tr>
<tr>
<td>8</td>
<td>68</td>
<td>91</td>
<td>30</td>
<td>84.8</td>
</tr>
<tr>
<td>9</td>
<td>109</td>
<td>143</td>
<td>35</td>
<td>80.0</td>
</tr>
<tr>
<td>10</td>
<td>117</td>
<td>208</td>
<td>40</td>
<td>72.8</td>
</tr>
<tr>
<td>11</td>
<td>135</td>
<td>280</td>
<td>45</td>
<td>64.2</td>
</tr>
<tr>
<td>12</td>
<td>108</td>
<td>350</td>
<td>50</td>
<td>53.4</td>
</tr>
<tr>
<td>13</td>
<td>77</td>
<td>406</td>
<td>55</td>
<td>44.5</td>
</tr>
<tr>
<td>14</td>
<td>64</td>
<td>436</td>
<td>60</td>
<td>36.9</td>
</tr>
<tr>
<td>15</td>
<td>44</td>
<td>434</td>
<td>65</td>
<td>30.4</td>
</tr>
<tr>
<td>16</td>
<td>32</td>
<td>400</td>
<td>70</td>
<td>24.8</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>340</td>
<td>75</td>
<td>20.5</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>265</td>
<td>76</td>
<td>17.3</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>189</td>
<td>73</td>
<td>15.0</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>122</td>
<td>66</td>
<td>13.2</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>15</td>
<td>24</td>
<td>11.5</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>5</td>
<td>13</td>
<td>11.2</td>
</tr>
</tbody>
</table>

4. Conclusion

We have proposed a system for off-line recognition of handwritten words containing Korean and numeric characters. Recognition accuracy of the system for word images on live mail pieces is 90.64% and the processing time for recognizing an image is 2.83 seconds. The major idea in implementing the system is to utilize the distinctive characteristics of Korean words and characters.

Acknowledgements: This work has been supported by University Research Program of Ministry of Information and Communication in South Korea.

References


