Lexicon Reduction Using Key Characters in Cursive Handwritten Words

Matthias Zimmermann
Institute of Computer Science and Applied Mathematics
University of Bern
CH-3012, Bern, Swizerland
zimmerma@iam.unibe.ch

Jianchang Mao (contact author)
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
USA
mao@almaden.ibm.com

Abstract The concept of *key characters* in a cursively handwritten word image is introduced and a method for extracting the key characters is presented. Key characters capture the unambiguous parts of the cursive words that can be reliably segmented and recognized. We also propose a method for lexicon reduction using key characters in conjunction with a word-length estimation. The presented method fits particularly well into cursive handwritten recognition systems which are based on over-segmentation followed by dynamic programming. Experimental results demonstrate the effectiveness of our method, yielding a lexicon reduction rate of 72.9% and a reduction of the lexicon matching time by 54.6% without increasing the error or the rejection rate.

Key words cursive script recognition, lexicon reduction, key characters, geometric constraints, length estimation.

1 Introduction

Hand-written word recognition is a challenging problem encountered in many real-world applications, such as postal mail sorting, bank check recognition, and automatic data entry from business forms.

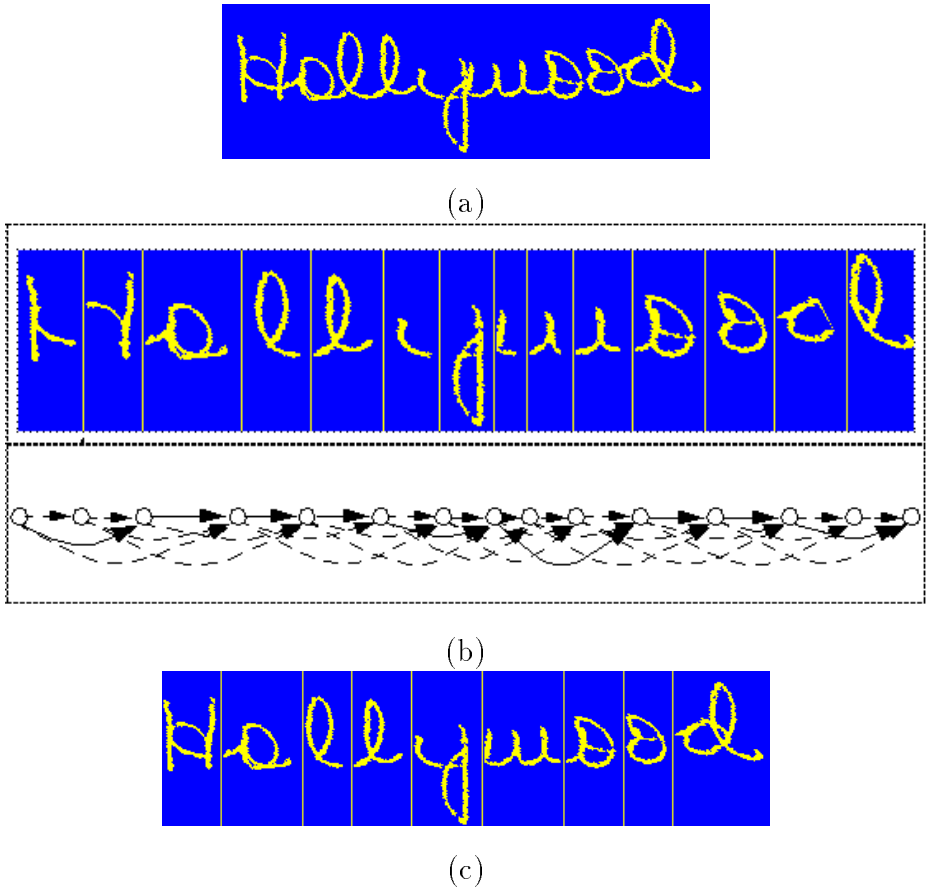
A prevalent technique for off-line cursive word recognition is based on *over-segmentation followed by dynamic programming* [1, 4, 8]. It seems to outperform segmentation-free Hidden Markov Models (HMMs) using a sliding window [9]. Over-segmentation based HMMs can also be built [2].

In the over-segmentation followed by dynamic programming approach, a set of split points on word strokes is chosen based on heuristics to divide the word into a sequence of graphemes (primitive structures of characters, see Figure 1(b)). A character may consist of one, two or three graphemes. Then, the word recognition problem is posed as a problem of finding the best path in a graph named *segmentation graph* (see Figure 1(b)). Each internal node in the graph represents a split point in the word. The leftmost node and rightmost node indicate the word boundary. Each edge represents the segment between the two split points connected by the edge. Since our over-segmentor rarely produces more than three graphemes for a character, we remove all the edges which cover more than three graphemes. A character classifier is usually used to assign a cost to each edge in the segmentation graph. The dynamic programming technique is then used for finding the best (hopefully the desired) path from the leftmost node to the rightmost node. A sequence of characters can then be obtained from the sequence of segments on the desired path (see Figure 1(c)).

Note that this sequence of characters may not form a valid word (or string) in a dictionary. Many paths in the segmentation graph can contain segments that are good-looking characters. Very often, the desired path may not appear in the top three paths selected by the dynamic programming algorithm. Therefore, in situations where a lexicon of limited size can be derived (e.g., in postal address recognition, a list of city-state names can be retrieved from a database once we know the zip candidates), a lexicon-driven matching is more desirable. For each entry in the lexicon, the dynamic programming technique is used to choose which path in the segmentation graph best matches with the entry, and a matching score is then assigned to the entry. The entry with the highest matching score is chosen as the recognition hypothesis.

Given a sequence of N graphemes and a string (lexicon entry) of length W, the dynamic programming technique can be used for obtaining the best grouping of the N graphemes into W segments [4]. A dynamic table of size $(N \times (W+1))$[1] must be constructed to obtain the best path. Mao, Sinha, and Mohiuddin [8] reduced dynamic table size to $(N-W+1)\times(W+1)$

[1]This is based on the notation used in this paper. In general, $(N \times W)$ is sufficient.



(a)

(b)

(c)

Figure 1: Over-segmentation followed by dynamic programming. (a) Original word image. (b) Graphemes and segmentation graph. (c) Segments on the correct path in the segmentation graph.

by taking advantage of the over-segmentation assumption (i.e., no under-segmentation). The reduction is significant when W is large (e.g., city-state-zip matching). Given a lexicon of L entries, the complexity of the lexicon-driven matching is $O(L \times (N - W + 1) \times (W + 1))$. As we can see from the above analysis, the speed of the lexicon-driven system decreases linearly with the lexicon size. Recognition accuracy also decreases when the lexicon size becomes larger. Therefore, it is very important to perform lexicon reduction in a lexicon-driven recognition system.

There are several techniques proposed in the literature for lexicon reduction [6, 7, 3]. Lexicon reduction based on holistic word features (word length, presence of ascenders, descenders, t-crossings, and i-dots) [10, 7] is commonly used. In this approach, holistic word features of the input image are matched against the holistic features of every exemplar for

each of the lexicon entries. Lexicon entries which do not match well with the holistic features of the input image are discarded. Typically, more than one exemplar must be stored or synthesized for each lexicon entry because of various different writing styles. Although the lexicon reduction based on holistic word features improves the overall speed of a system by reducing the lexicon size for final recognition, its efficiency is limited by the computational overhead for extracting holistic word features and feature-matching with more than one exemplars for each lexicon entry.

Kimura et al. [5] proposed a lexicon reduction method in which the input image is first segmented into segments based on a set of heuristics and an ascii string is created based on the recognition results on these segments. A dynamic program is then used to match this ascii string with each lexicon entry. Lexicon entries with high matching costs are eliminated from further consideration. The performance of this approach heavily relies on the initial segmentation of word into "characters" which is problematic for cursive words.

In this paper, we define the concept of *key characters* in a cursive word image and present a method for extracting the key characters. The key characters capture the unambiguous parts of the cursive words that can be reliably segmented and recognized. We also propose a method for lexicon reduction using key characters in conjunction with a word-length estimation. One advantage of this method is that it fits well into any cursive handwritten recognition system based on over-segmentation followed by dynamic programming. The overhead in extracting key characters and length estimation is very small. Unlike in the method by Kimura et al. [5] which requires segmenting the words twice, the proposed method makes use of the available graphemes. Our proposed method is also more reliable than the method by Kimura et al. [5] because the former does not use the unreliable segments and their recognition in matching with the lexicon entries.

This paper is organized as follows. In section 2 the concept and the extraction of key characters is described. Section 3 explains our approach to estimate the length (i.e. the number of characters) of a given image and section 4 explains the lexicon reduction using both key characters and length estimation. Experiments and results are reported in section 5 and Conclusions are made in section 6. All graphical illustrations are based on the address line shown in Fig. 2 where graphemes are identified by boxes.

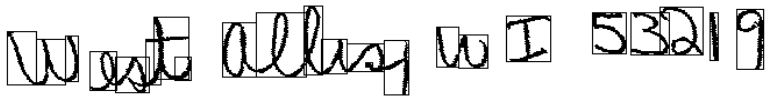


Figure 2: Graphemes in an address line.

2 Key Characters

Key characters identify unambigous characters of cursive words which can be segmented and recognized reliably without performing word recognition or contextual analysis. The extraction and recognition of key characters works directly on the sequence of graphemes which can be obtained by any oversegementation method (see Fig. 2 for the graphemes obtained by the system described by Mao et al. [8]).

Generally it is not reliable to identify characters in cursive words solely based on recognition of individual segments due to the following facts: (i) The graphemes of many characters can be "good-looking" characters themselves. (ii) Combination of neighboring graphemes (from the same characters or from different characters) can be "good-looking" characters too.

On the other hand, our observation on a large number of cursive words indicates that there often exist some parts in cursive words that can be reliably segmented and recognized. This observation leads to a formal definition of *key characters* in a cursive word.

Let $G = (GR_1, GR_2, \ldots GR_n)$ be the sequence of graphemes (GRs) in a given image and S_{ij} be a segment consisting of a subsequence in G: $S_{ij} = (GR_i, GR_{i+1}, \ldots GR_{i+j-1})$; $i \in [1, \cdots, n], j \in [1, \cdots, m]$ and $i + j - 1 \leq n$ where m is the maximum number of GR's which are allowed to form a character (in our case $m = 3$).

Let $ch_x(S_{ij}) \in [0, \cdots, 100]$ be the output of a character classifier which computes a confidence for S_{ij} to represent a character x. Let $m(S_{ij}) = \max_x(ch_x(S_{ij}))$ be the maximum output of the classifier for the segment S_{ij}. The definition of a key character candidate can now be formulated as follows:

Definition: A segment S_{ij} represents a *Key Character* or *KC*, if the following three conditions are met:

(i) $m(S_{ij}) > T$;

(ii) $m(S_{ij}) > m(S_{kl}) + d$, $\forall S_{kl}$: $S_{ij} \cap S_{kl} \neq \emptyset$ if $S_{kl} \neq S_{ij}$ for a pre-specified $d \in [0, \cdots, 100]$;

(iii) S_{ij} must satisfy certain geometric constraints.

The threshold T controls the minimum confidence that a KC must have. The value d should be selected in proportion to the requirement of how much 'better' the KC has to be compared to its environment (the S_{kl}).

Intuitively speaking, a segment is said to be a KC if no other segment which contains at least one grapheme in this segment can claim themselves as a KC. Therefore, this definition of KCs captures the unambiguous parts of a cursive word.

Let $A = A_1, A_2, \ldots A_q$ denote the set of all KCs for a given G. A has the following property: No two KCs in A share one or more GRs, i.e., $A_i \cap A_j$, $i \neq j$. This is a nice property because all the KCs can be valid simultaneously in a cursive word.

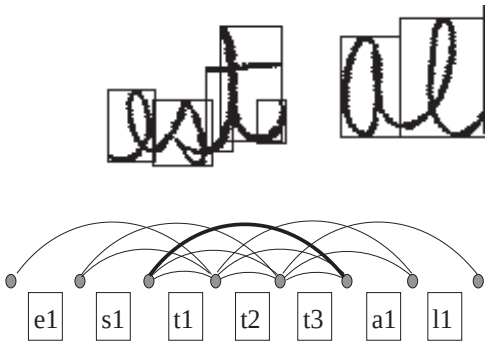


Figure 3: Selecting key character candidates.

Based on this definition, we implement a method for extracting key characters in a cursive word. The method has two steps which are explained in the following two subsections. In step one, a set of key character candidates which satisfy conditions (i) and (ii) in the definition are extracted.Then, these key character candidates (KCCs) are screened by their geometric properties in the text line.

2.1 Key Character Candidates

Fig. 3 shows the selection of the character 't' as a KCC which is represented by segment $S_{6,2}$ (see Fig. 2). This is indicated by the bold arc, the other arcs represent all S_{ij} which share one or more GRs with $S_{6,2}$.

Fig. 4 shows all extracted KCCs for the sequence of GRs in Fig. 2. Tab. 1 lists the top three interpretations x with their assigned confidences for each KCC ($T = 70$, $d = 10$). Only characters x with $ch_x(S_{ij}) \geq 5$ are listed.

Figure 4: Key character candidates with bounding boxes.

index	1	6	9	13	17	18	19	20	21	22
length	3	3	1	2	1	1	1	1	1	1
1st	W (92)	T (99)	A (95)	4 (84)	I (88)	5 (100)	3 (100)	2 (100)	1 (99)	9 (100)
2nd				Y (11)	5 (45)	S (79)	S (96)	D (51)	L (83)	G (91)
3rd					1 (13)	G (21)		A (8)	I (13)	

Table 1: Key character candidates with character recognition results.

In the first row of the table, the index of the first GR in each KCC is given and the second row indicates the number of GRs for each KCC. In this example all the top choices of the character classifier represent the correct interpretations. The observation of hundreds of KCCs showed that the correct interpretation of the KCC can most often be found in the top two choices.

2.2 Screening KCCs Using Geometrical Properties

In order to achieve a high confidence that the KCC are correctly segmented from their background and are correctly recognized, a verification step applies additional constraints to the KCCs.

The goal of the verification process is to select a subset of the KCC to form a reliable set of KCs. This is done in two steps. The first step removes all KCCs which contain characters in the top three interpretations which correlate often with character substitutions or segmentation errors. The following characters have been found to correlate most often with substitution errors: 'l', '1', 'r', 'o', 't', 'f' and '2'.

The second verification step is more complex. It considers character class specific geometrical properties of the KCC, such as the normalized vertical position and the normalized height of a KCC.

Estimating the Base and Ascender Line

For both the normalized vertical position and the normalized height of a KCC, it is necessary to estimate a base and an ascender line (Fig. 8 shows estimated base and ascender lines). The estimation of the base line is done by simple line fitting using the least square method for the bottom left corners of the bounding boxes of the extracted GRs. The coordinates of the most distant corner are iteratively removed for the final computation of the base line.

For the ascender line the coordinates of the top left corner of the GRs are used in the same way with the exception that some of the coordinates are modified to comply with small letters. The height of upper case characters is roughly estimated by the average of the mean and the maximum height of all GRs. The coordinates of all GRs with a smaller height are then adapted to this estimated height for upper case characters before the line fitting is applied.

Geometrical Properties

In addition to the average number of transitions of a KCC, its normalized height and normalized vertical position are used for the geometrical verification.

The normalized position pos_{norm} is defined as the distance from the estimated base line to the middle of the bounding box of the character pos_{ch} divided by the distance between the base and the ascender line h_{line} measured at the position of the KCC. The normalized height h_{norm} is defined as the height of the bounding box of the character h_{ch} divided by h_{line} (see Fig. 5 for the definition of the mentioned quantities).

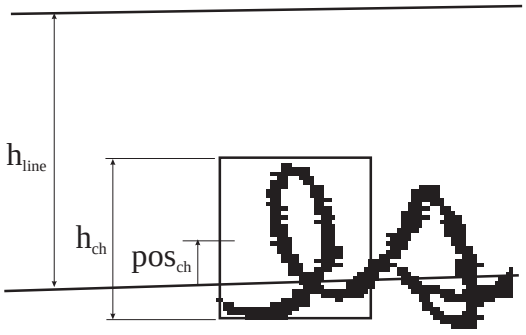


Figure 5: Size and position of a character.

Using training data which provides the truth value associated to each GR, the distribution of the geometric properties can be estimated for each character class. Fig. 6 shows two histograms for the character specific distribution of geometrical properties. On the left side

the histogram for the mean number of transitions for the character class 'W' is presented and the normalized height for the character class 'F' is shown on the right side.

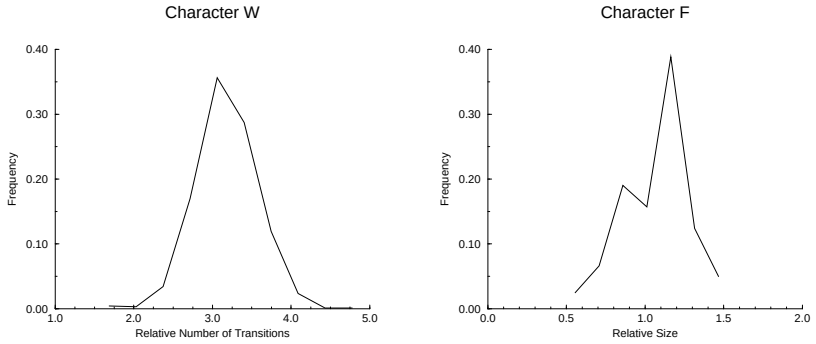


Figure 6: Histograms for transitions and height.

The transition histogram indicates that in most cases the normalized number of horizontal transitions is close to three according to the sampled writing styles provided by the training data. Since the current implementation treats capital and small letters as the same character class the histogram for the letter 'F' shows two peaks one for capital (normalized height of 0.8) and one for small letters (normalized height of 1.2).

Geometrical Confidences

The idea of a geometrical confidence is to provide information about how 'good' a certain value for a given geometric property is. It is defined as the probability for not having a better value. Given the density function d_{ix} of a geometric property i and a character class x the geometrical confidence $conf_{ix}(v)$ for a given feature value v is defined as follows

$$conf_{ix}(v) = \int_{-\infty}^{+\infty} d_{ix}(u) I(d_{ix}(u) \leq d_{ix}(v)) du, \tag{1}$$

where $I()$ is an indicator function. According to this definition a value $conf_{ix}(v) \in [0,1]$ is produced. It is motivated by the observation, that 'good' feature values v lead to a high $d_{ix}(v)$. Fig. 7 illustrates this concept graphically.

For the desired geometrical verification of the KCCs, the confidences are computed for each KCC and each property. In order to be accepted as a KC a KCC has to achieve the following minimum confidences: 0.5 for the mean number of transitions and 0.2 for both the normalized vertical position and the normalized height.

Fig. 8 shows the KCs which remained after the verification process for the extracted KCCs in Fig. 4. The character 'T' and the digits '2' and '1' have been removed in the

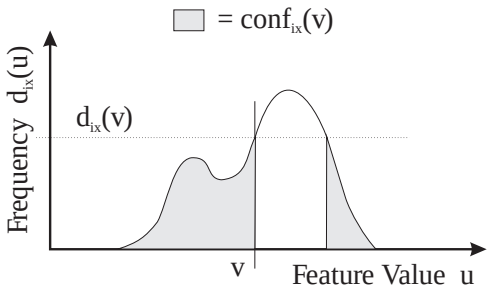


Figure 7: Geometrical confidence.

first verification step. The geometric verification excluded the letter 'S' (combined with the comma) and the letter 'I'.

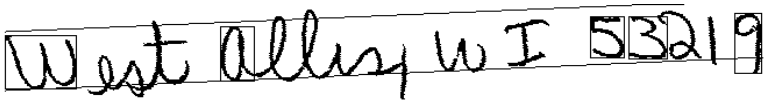


Figure 8: Key characters with boxes.

3 Length Estimation

The goal of length estimation is to provide an estimate of how many characters are present in a given image without performing expensive recognition. Given an image of text line, five features are first extracted from the line. A neural network is used to estimate the line length (in term of the number of characters). Since such an estimation will not be accurate without recognition, a range for the length is also provided by the network.

Since in our system the image is already segmented into connected components (CCs) and graphemes (GRs) the number of CCs and GRs can be directly used as the first two features without any additional computation.

Additional information for the lenth estimation can be extracted by measuring the number of horizontal transitions in the image. The number of horizontal transition is given by the number of locations where a black pixel on a scan line is followed by a white pixel. The third feature is the average number of horizontal transitions per scan line, $trans(image)$. The fourth feature is similar to the third. Instead of measuring the value $trans(image)$, it uses the sum of $trans(GR_i)$ over all GRs. The fifth feature is the average height of the GRs (normalized by the maximum height found in the provided GRs).

feature	v	μ	σ	remarks
f1	12	5.64	4.50	number of CCs
f2	22	8.31	7.09	number of GRs
f3	16.23	5.91	3.83	normalized transitions
f4	31.11	11.25	9.60	sum of normalized transitions per GR
f5	0.66	0.72	0.17	average normalized height of GRs

Table 2: Selected features for length estimation.

Tab. 2 shows the extracted feature values for the city state zip line in Fig. 2. The mean (μ) and standard deviation (σ) of each feature are also shown in Tab. 2. They are estimated from a training set.

Each of the five features are normalized to have a zero-mean and unit variance. The normalized features are then fed to a feedforward neural network with five input units, 10 hidden units, and three output units. The three output units produce estimates of the best length (len_{best}), maximum length (len_{max}), and minimum length (len_{min}), respectively.

The network was trained on a set of truthed data of 5000 lines and partial lines (between neighboring key characters). Each line or partial line is associated with the true number of characters. An independent data set of 3200 lines was used for testing. The standard back propagation learning was used for the training of the len_{best} output neuron. After a few epochs (less than 10) stable recognition rates of 57% on the validation set were achieved.

We propose the following approach to estimate the minimum and the maximum number of characters present in a given image. The two output units corresponding to the minimum len_{min} and the maximum length len_{max} have to be trained slightly different from the len_{best} output unit. Instead of using the difference $delta = len_{true} - len_{best}$ in the back propagation step, the difference is scaled up or down depending on whether the length is under-estimated or over-estimated by the len_{best} unit. In the current implementation, the new $delta'$ for the len_{min} unit is computed as follows:

$$delta' = \begin{cases} a\,delta & \text{for } delta < 0 \\ b\,delta & \text{for } delta \geq 0 \end{cases}$$

where $a = 8$ and $b = 0.25$. By exchanging the values for a and b, the same formula is used

for training the len_{max} unit. The network trained using this scheme produces an average range of length 5.98 and a coverage of 99% (the true length was found within the range).

4 Two-Stage Lexicon Reduction

We implement a lexicon reduction method which consists of the following two stages. (i) Lexicon reduction using length estimation, and (ii) Lexicon reduction using key character strings. The order of these two steps was determined by the fact that examining an lexicon entry in the frist stage is faster than in the second stage.

4.1 Lexicon Reduction Using Length Estimation

Once the estimation for a minimum and a maximum number of characters is available the two numbers can be used to find lexicon entries which do not fit into the provided range. For the city state zip line in Fig. 2 the estimated range for the true length is from 11 up to 18 characters.

4.2 Lexicon Reduction Using Key Character Strings

Once the key characters are extracted, a search string can be generated by combining length estimation and KCs. The set of all possible search strings can be defined by the following regular expression: $(A^*-^*)^+$ where A represents the set of the selected number of top choices made by the character classifier and '-' represents a wild char. The number of the wild chars between two subsequent KCs is estimated using the same neural network as for the length estimation of the address lines containing the city state zip information. Applied to the GRs in Fig. 2 this procedure leads to the following search string `{W}--{A}------{5,S}{3,S}--{9,G}`. If only the top choice for each KC of the character classifier is represented the search string can be written as `W--A------53--9`.

Since the number of wild chars, the position of the KCs in the search string and the correct interpretation do not necessarily correlate with the truth value, a flexible matching (allowing a certain amount of mismatches) for search strings and lexicon entries has to be applied. If the search string contains n KCs, the maximum number of allowed mismatches is set to $1 + n/4$. Therefore no further lexicon reduction is performed if the generated search

string contains less than two KCs.

Tab. 3 demonstrates the matching process of a search string and lexicon entry. For each KC in the search string a search position and a search range within the lexicon entry is defined depending on the position of the KC within the search string. Since the length of the lexicon entry and the search string are in most cases not the same the search position in the lexicon entry is set to the same relative position as the one of the KC in the search string. The minimum search range is set to ± 1 at the beginning and at the end of the lexicon entry and grows linear towards the middle of the lexicon entry up to the maximum search range of ± 2 character positions.

lexicon entry	+EST+LLISWI53219
search range	.....[x]...
search string	W--A------53--9
	..........SS..G

Table 3: Matching of a key character.

In Tab. 3 the current search position is marked by the letter 'x' and the search range by square brackets. It can be seen that the first two KC have been successfully matched (marked with the plus sign to prevent that a single character of the lexicon entry is used more than once). At the current search position the character '5' of the lexicon entry will be marked with the plus sign since the system tries first to find a match for the top choice of the KC.

Finally the number of mismatches is computed by subtracting the number of plus signs from the number of KCs in the search string and tested against the allowed maximum of mismatches for the current search string.

5 Experiments and Results

This section presents the results for both the lexicon reduction and the overall system performance of the recognition system for handwritten addresses.

5.1 Data for Training and Testing

All the experiments were done using images provided by United States Postal Services (USPS). In order to generate the character statistics for the geometric confidences and to extract the length estimation features for the training of the neural network truthed data of about 800 images was used. For the testing of the lexicon reduction 811 addresses have been used which include not more than 175 images from the training data.

5.2 Lexicon Reduction Performance

This subsection presents the lexicon specific performance measurements. The lexicon reduction using length estimation reduced the lexicon size on the average by 54.4%. The search string matching achieved a further reduction of 37.5% which resulted in an total average reduction of 72.9%. In 98.6% of the cases the truth value was preserved during lexicon reduction and the relative amount of time saved for the lexicon matching (including the reduction step) was 54.6%.

5.3 System Performance

Tab. 4 summarizes the results for the measurements of the overall speedup and Tab. 5 presents the corresponding results for the measurements of the rejection rate and the error rate. The statistical data for both tables were collected from two experiments (with lexicon reduction switched on and off).

red	t	t_m	t_r	Δ spd
off	2033.11	252.12		
on	1779.14	64.57	49.77	12.5%

Table 4: Execution times in milli-seconds with and without lexicon reduction.

The first column `red` indicates whether the lexicon reduction has been switched on or off. The total execution time for each experiment is given by t using the specified test set. The column t_m reports the used time for the lexicon matching phase and t_r gives the time needed to reduce the lexicon size.

red	imp	acc	rej	rec	sub	Δ rej	Δ err
off	740	684	56	641	43		
on	740	682	58	640	42	3.6%	-2.3%

Table 5: Rejection and error rates with and without lexicon reduction.

The heading `imp` stands for the number of processed images and the number of rejected and recognized addresses are in the columns `rej` and `rec`. The number of the cases containing errors in the 5 digit zip code are shown in column `err`. The effect of the lexicon reduction on the rejection and the error rate are given in last two columns Δ `rej` and Δ `err`. These final results show that the lexicon size could be reduced by over 72% gaining an overall speed up of over 12% without changing the rejection rate (+3.6%) and the error rate (-2.3%) significantly.

6 Conclusions

We introduced the concept of key characters which capture the unambigous parts of a cursive word image. Key characters can be reliably extracted and recognized without performing word recognition or contextual analysis.

The proposed approach of lexicon reduction can be used for speeding up any handwriting recognition system which is based on over-segmentation followed by a lexicon matching step. Only minor changes of an existing system are necessary to achieve a significant speed up without an increase of the error or the rejection rate.

Key characters in cursive words can alse be used for coarse recognition of cursive words which can be followed by any other fine recognition methods.

Key characters in cursive words can further be used for improving the dynamic matching of graphemes with an lexicon entry by weighing more on the key characters. This adaptive weighting approach can be extended to lexicon-free recognition to increase the probability that the "best path" in the segmentation graph corresponds to the desired (or true) path.

References

[1] R. M. Bozinovic and S. N. Srihari. Off-line cursive word recognition. *IEEE Trans, PAMI*, 11:68–83, Jan 1989.

[2] M. Y. Chen, A. Kundu, and S. N. Srihari. Variable duration hidden markov model and morphological segmentation for handwritten word recognition. *IEEE Trans. on Image Processing*, 4(12):1675–1688, December 1995.

[3] M.Hadorn G. Kaufmann, H. Bunke. Lexicon reduction in a hmm-framework based on quantized feature vectors. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 1097–1101, Ulm, Germany, Aug. 1997.

[4] P. D. Gader, J. M. Keller, R. Krishnapuram, J. H. Chiang, and M. A. Mohamed. Neural and fuzzy methods in handwriting recognition. *Computer*, pages 79–85, February 1997.

[5] F. Kimura, M. Shridar, and N. Narasimhamurthi. Lexion driven segmentation - recognition procedure for unconstrained handwritten words. In *Proceedings of International Workshop on Frontier in Handwriting Recognition*, pages 122–131, Buffalo, NY, 1993.

[6] S. Madvanath and V. Govindaraju. Holistic lexicon reduction. In *Proceedings of International Workshop on Frontier in Handwriting Recognition*, pages 71–81, Buffalo, NY, 1993.

[7] S. Madvanath and S.N. Srihari. Effective reduction of large lexicons for recognition of offline cursive script. In *Proceedings of International Workshop on Frontier in Handwriting Recognition*, pages 189–194, Essex, England, 1996.

[8] J. Mao, P. Sinha, and K. Mohiuddin. A system for cursive handwritten address recognition. In *Intl. Conf. on Pattern Recognition. Brisbane, Australia*, Brisbane, Australia, Aug. 1998.

[9] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. Pattern Anal. Machine Intell.*, 18(5):548–554, May 1996.

[10] S. N. Srihari. Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern Recognition Letters*, 14:291–302, April 1993.