Segmentation-free Keyword Spotting for Bangla Handwritten Documents

Xi Zhang
School of Computing
National University of Singapore
13 Computing Drive, 113417, Singapore
xizhang@comp.nus.edu.sg

Umapada Pal
Computer Vision And
Pattern Recognition Unit
Indian Statistical Institute, Kolkata, India
umapada@isical.ac.in

Chew Lim Tan
School of Computing
National University of Singapore
13 Computing Drive, 113417, Singapore
tancl@comp.nus.edu.sg

***Abstract*—In this paper, a segmentation-free keyword spotting method is proposed for Bangla handwritten documents. In order to tolerate large variations in handwritten scenarios, we extracted keypoints based on SIFT keypoint detector, and the end and intersection points found by morphological operations. Heat Kernel signature (HKS) is used to present the local characteristics of detected keypoints. Instead of using the same size of patch for all the keypoints, we apply a method dynamically deciding the patch size. Furthermore, our spotting method reduces the scope of searching on the document by only considering the candidate local zones with similar candidate keypoints, and does not need pre-processing steps. From the experiment on Bangla handwritten text we obtained encouraging results.**

I. Introduction

Nowadays, most of valuable documents are scanned into Digital Libraries and databases, and public services are provided for information retrieval based on the user requirements. Optical Character Recognition (OCR) is one of the most widely used method, trying to convert the imaged documents into text format, by recognizing segmented characters or words. However, due to degradation of low-quality historical manuscripts, and large variations in handwritten documents, OCR may have poor recognition results.

Keyword spotting [1] is an alternative way to index and retrieve useful information, without knowing the ASCII of content. Geometrical column features are extracted from segmented word images, and Dynamic Time Warping (DTW) is applied to calculate the similarity between two sequences of features and return a list of scored word images with respect to the query image. Therefore, keyword spotting is a content based method, other than recognition. However, due to variant writing styles or different individuals, pre-processing steps including binarization, skew and slant correction, and normalization, are very important, and the extracted features should be robust and invariant to these variations. Moreover, because of unconstrained writing styles, consecutive characters may be connected together, and long ascenders and descenders lead to difficulty in text line segmentation. Thus, many methods nowadays focus on segmentation-free methods for keyword spotting.

One kind of methods use supervised learning models directly on the segmented text lines to avoid word segmentation errors. In [2], Hidden Markov Models (HMM) are trained for each character on text lines in the training set, and can output the character sequence with the highest probability for every testing text line. In order to spot the query word, rather than recognition, using a special defined language model, the trained network can return which text lines contain the query word with higher probabilities. However, HMM suffers from some drawbacks, as described in [3]. Recurrent Neural Network (RNN) with Bidirectional Long Short-Term Memory (BLSTM) hidden nodes and Connnectionist Temporal Classification (CTC) output layer achieves better keyword spotting results [4]. However, HMM and RNN need a large amount of training data, and pre-segmented text lines.

Another kind of methods avoids any segmentation thoroughly, an example of which was proposed in [5]. The authors presented a language independent word spotting method for ancient manuscripts without segmentation. Only informative parts on the documents and the query are used to retrieve the query word, which are named as guides, and the zones of interest (ZOI) are extracted from the query, which defines the truly informative and is stored in a tree in order to construct the geometrical structure. The features of both the guides and ZOIs are based on gradients. Then, the first ZOI in the query is matched to every guide on the document, and a cohesive matching process is used to calculate the matching score.

In [6], based on the size of the query word image, Scale-Invariant Feature Transform (SIFT) [7] features are extracted from similar size patches, which are sampled densely on the document. Using the feature vectors of all patches, Latent Semantic Indexing is applied to transform the feature space of the document to the topic space, so that when we project the feature vector of the query image to the topic space of one document, we can get the relevance between the query image and the document using cosine distance measure, and in a voting space, patches with enough evidences are returned as the spotting results.

However, features based on gradients may not enough to tolerate large variations of unconstrained writing styles, such as the non-rigid deformations in real scene images [8], moreover, matching the query for every appearing guides or sampling the patches densely may lead to large searching effort. In this paper, we propose a segmentation-free method for spotting query word images in Bangla handwritten documents. Heat Kernel signature (HKS) is used to present the local characteristics of detected keypoints, which is proven to be invariant to the non-rigid deformations much better that some other features in [8]. Furthermore, only the local zones with similar candidate keypoints are matched to the query image, in order to reduce the searching scope. The local zones with enough matching keypoints are sorted by the matching scores and construct the spotting list.

The rest of the paper is organized as follows. Section II discusses about descriptor generation and Section III details about keyword spotting techniques. Experimental results are given in Section IV. Finally, the paper is concluded in Section V.

II. Descriptor Generation

A. Heat Kernel Signature

Given a handwriten image, we can extract the Heat Kernel Signature (HKS) of all the keypoints using the methods described in [9]. Because we use the binarized documents in the experiments, we only calculate the HKS descriptors, instead of using its scale invariant version.

B. Localization of Keypoints

In document images, regardless of printed or handwritten documents, the important information we can use to retrieve useful information is the characteristics of the appearing characters or words. Densely extracting features from all points on the strokes is very time consuming, not only for extracting and storing the large features, but also for similarity matching. Therefore, how to localize the important keypoints, which can capture the maximum information the documents can provide, is one of the most crucial works we should consider.

There are four normally used keypoint detecting algorithms: Harris algorithm [10], SIFT (Scale Invariant Feature Transform) keypoint detector [7], LoG (Laplacian of Gaussian) algorithm [11], and Morphological operation (MO) on binary images [12]. In Fig. 1, the keypoints detected by these four algorithms are shown. We can see that in Fig. 1(a), Harris always find keypoints along the boundary of strokes, and SIFT in Fig. 1(b) detect some keypoints in the background, which are not useful. In Fig. 1(c), all the keypoints are along the strokes, but some keypoints are located very near to each other, the features of which may be very similar, leading to duplication. The most important characteristics the strokes contain are always located at the start or end of strokes, intersections of strokes, or the positions where the strokes try to change its orientation, so the end points and intersect points by morphological operations in Fig. 1(d) are always the desirable points we want. However, in some cases, some keypoints may be missed. Therefore, in order to include all the keypoints we want, we combine the results of SIFT and MO.

For the keypoints detected by SIFT detector, we remove the keypoints in the background, by checking the intensity values. If the images are binary (1 is the foreground, 0 is the background), we can easily remove the keypoints with the intensity value of 0, as shown in Fig. 2(a). After combining the keypoints with MO, as shown in Fig. 2(b), we can find out that some keypoints are located in the very near positions. Therefore, we group keypoints with small Euclidean distances to each other together, and only keep one of them, which is located nearest to their center. The final keypoints we use in our experiments are shown in Fig. 2(c).

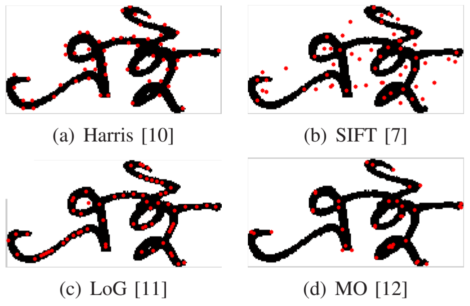

(a) Harris [10] (b) SIFT [7] (c) LoG [11] (d) MO [12]

Fig. 1. Keypoints detected by different algorithms.

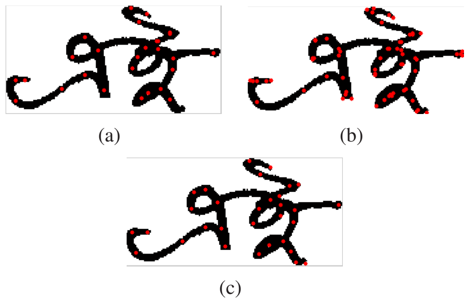

(a) (b) (c)

Fig. 2. The final Keypoints we will use in the experiments. (a) Keypoints detected by SIFT detector after removing keypoints in the background. (b) Combining the keypoints in Fig. 2(a) with the ones in Fig. 1(d). (c) Removing near keypoints.

C. Size of Local Patch

The descriptors are extracted from the local patch centered at each keypoint we detect, because in handwritten documents, the same word may be written in variant writing styles, fixing the size of all patches may bring in unwanted parts of strokes, so that the descriptors of the similar keypoints in different occurring same words may have large difference.

In order to automatically decide the size of patch centered at one keypoint, we use the Entropy values in different sizes of patches centered at the same keypoint to find the optimal one. [5] also used the entropy to compute the size of ZOI (zones of interest). Centered at a keypoint kp, we calculate HKS descriptor HKS(kp), from a patch P with the size of $(2*r+1)\times(2*r+1)$, where $r\in[5,50]$. For each r, we get the Entropy as following equations:

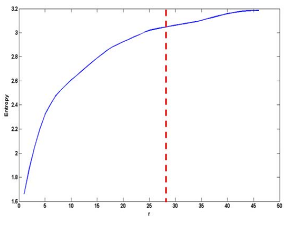
$$Energy = \text{HKS}(kp, t=0) \tag{1}$$

$$Entropy_r = -\frac{1}{E}\times\sum_{p_i\in S} Energy(p_i) * ln\frac{Energy(p_i)}{E} \tag{2}$$

where S is the set of all points on the strokes in P, $Energy(p_i)$ is the Energy value of p_i, and $E=\sum_{p_i\in S} Energy(p_i)$.

While r is increasing, more points are included in S, so that the energy becomes large, so does the entropy value. However, the increase of entropy values is large at the beginning, but slows down when enough information is included. Fig. 3(a) shows the entropy values of the patches centered at the left most keypoint in Fig. 2(c), when r is increasing from 5 to 50.

We can see that at the beginning, the entropy value increases very fast, but slow down when r is bigger than 20. In order to track when the entropy starts slowing down, namely, the patch has already include enough information, we calculate the gradient values shown in Fig. 3(b), and choose the position when the gradient begins smaller than ϵ. If we choose $\epsilon = 0.02$, the optimal size of the patch is 23, as indicated as the red dot lines in Fig.3.


(a) Entropy values.

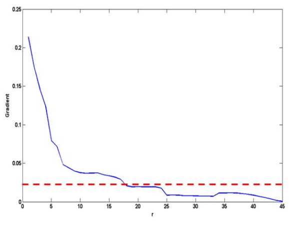
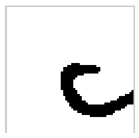
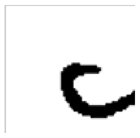

(b) Gradient of Entropy values.

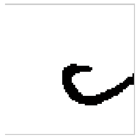
Fig. 3. Illustration of sizing the local patch

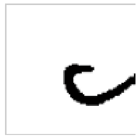
D. Patch Normalization

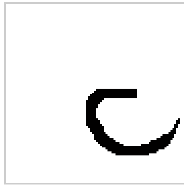
The similarity measure of the distance between two HKS descriptors requires that the descriptors have the same dimension, however the size of patches may be different, so we should normalize all the extracted patches to be the same size and then calculate the HKS descriptors. If we directly resize all the patches to a predefined size, the width of strokes will vary differently based on the size of original patches. As shown in Fig. 4, Fig. 4(a) and Fig. 4(c) are the two patches centered at the same keypoint, but with different r, if we resize these two patches to the same size 91×91, in Fig. 4(b) and Fig. 4(d) respectively, the width of strokes are different, obviously the stroke in Fig. 4(b) is thicker.


(a) $r = 30$


(b) 91×91


(c) $r = 40$


(d) 91×91


(e)

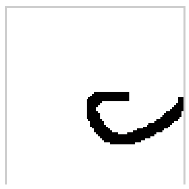

(f)

Fig. 4. Resizing patches with different r to the same size will lead the width of stokes different. (e) Normalize the patch in 4(b). (f) Normalize the patch in 4(d).

In order to keep the width of strokes unchanged, we first apply the thinning method [13] on the strokes in the resized patch, so that the width of strokes is 1, and then remove the spurious pixels to smoothen the strokes. We then apply dilation twice, and at last only keep the foreground pixels in a circle with the same radius as the side length of the patch, in order to be invariant to the rotations. The normalized patches of Fig. 4(b) and Fig. 4(d) are shown in Fig. 4(e) and 4(f) respectively, and the widths of the strokes are the same.

Therefore, for every keypoint, we get the local patch with the size determined by the method in Section II-C, and normalize all the patches to the same size, so that, descriptors are all in the same dimension. Based on the descriptors, in the next Section, we will introduce how to spot the locations where the query word appears on the document.

III. Keyword Spotting

A. Candidate Keypoints

Given a query image and a document, we should first search throughout the whole document, and find the possible matching areas. In order to relief the effort of checking all the parts of the document, such as densely moving a sliding window, we only consider the areas containing similar keypoints, with respect to the ones in the query image. Therefore, if we assume that the query image has n_q keypoints, denoted as kp_i, $i \in [1, n_q]$, and the document has n_d keypoints, denoted as kp'_j, $j \in [1, n_d]$, we first calculate the distance between every pair of kp_i and kp'_j by the following equation:

$$d_i^j = d(kp_i, kp'_j) = \| \text{HKS}(kp_i), \text{HKS}(kp'_j) \| \qquad (3)$$

where $\| \cdot \|$ is the L2-norm.

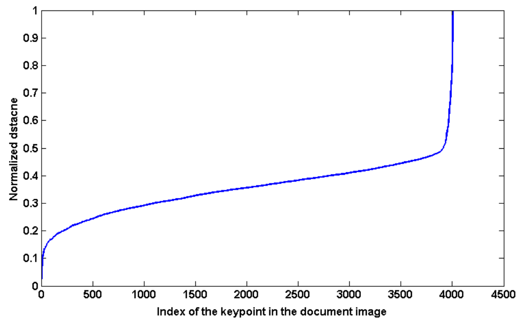


Fig. 5. An example of the sorted distances of one keypoint in the query image with respect to all the keypoints in one document.

For every kp_i, we can get a list of distances d_i^j, $j \in [1, n_d]$, and then all the distances are normalized by scaling to $[0, 1]$. Fig. 5 shows an example of the sorted distances between the descriptor of one keypoint in query image and the descriptors of all the keypoints in one document. We can see that the distances increase very fast at the beginning, and then slow down for most of the keypoints. At last, the distances increase dramatically. Therefore, we can divide the change of the sorted distances into three parts: the first part contains most of the similar keypoints with respect to the one in the query image, the second part contains the keypoints with few small similar parts, and the third part contains most of the dissimilar keypoints. In order to get the candidate keypoints, we only consider the first part, so that the keypoints in the first parts and with the distances in the top 20% smallest are considered as the candidate keypoints. Using this threshold method, different keypoints in the query image may have different numbers of candidate keypoints on a document, because the patch centered at a keypoint always contains a part of a character, which can also be a part of other different characters, and moreover, the number of occurrences of different characters are different due to specific language or different content on the document, some patches may appear much fewer times than the others. Our threshold method try to capture all the similar patches, but bring in the dissimilar patches as few as possible.

Candidate keypoints for one keypoint kp_i in the query image is denoted as $Candi(kp_i)$, and we also record the coordinates of the candidate keypoints, which will be used in the next section to locate the candidate matching zones. $M(kp'_j)$ stores the keypoints in the query image, to which kp'_j is similar, namely, kp'_j is the candidate keypoint of every keypoint in $M(kp'_j)$. In addition, a matrix $Mark$, with the same size of the document is used to record the positions where the candidate keypoints appear, by setting the values to 1, and others to 0.

B. Localization of Candidate Local Zones

Instead of densely moving a sliding window throughout the whole document, we try to reduce the searching effort, and focus on the areas with candidate keypoints.

Assume the size of the query image is $m \times n$, and the coordinates of the keypoints kp_i in the query image is denoted as (x_i, y_i). For each kp_i, the candidate local zones are defined as the windows with the bounding box of $((x'_j - x_i, y'_j - y_i), m, n)$, which denotes the coordinate of the left-bottom position, the height, and the width, and (x'_j, y'_j) are the coordinates of $kp'_j \in Candi(kp_i)$. In order to avoid missing any candidate local zones, we extract windows for all kp_i, because we can not grantee that all the similar keypoints can be included in the candidate keypoints, if we only consider the candidate keypoints of subset of kp_i, some positive matching zones may be missed, as shown in Fig. 6. Fig. 6(a) shows the candidate local zones for the first keypoint in Fig. 2(c), and some positive matching area are not included. However, in Fig.6(b), which shows the candidate local zones of the second keypoint, some missing areas are spotted. Therefore, using the candidate local zones for all the keypoints in the query image can spot as many positive matching areas as possible.

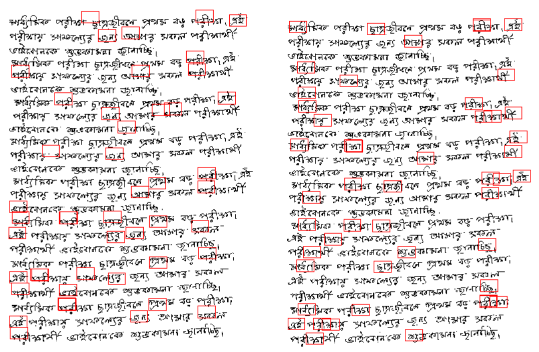

(a) Candidate local zones of the first keypoint in 2(c).
(b) Candidate local zones of the second keypoint in 2(c).

Fig. 6. Illustrations of candidate local zones

C. Matching Score

Given a candidate local zone, the matching score is calculated based on the candidate keypoints inside. Based on the bounding box of the candidate local zone, and the matrix $Mark$, we can get a set of keypoints on the documents, denoted as LZ, each of which is at least a candidate keypoint of one keypoint in the query image, and all of which are sorted by their vertical position values. Our aim is to get a matching path in the candidate local zone from left to right, so we use two variables to record the matching process: $Hist$ storing the matching path, and $Score$ storing the optimal matching score. Our proposed method for calculating the matching score is shown in Algorithm 1.

For each kp'_j in LZ, we first store $M(kp'_j)$ in one column of MS, and then initialize the first column $Hist$ to $MS(LZ[1])$, storing the corresponding matching distances to the first column of $Score$. Fig. 7 shows an example of MS, each column of which are the indexes of the keypoints in $M(kp'_j)$, such as in the second column, $LZ[2]$ is the candidate keypoint of the 3rd, 8th, and 16th keypoint in the query image. $Hist$ and $Score$ always have the same size of MS. For each cell of MS, we check all the cells in the several previous columns, for example, if the first cell in the forth column of MS in Fig. 7 marked as light gray, we check all three previous columns in the dark gray. Because some keypoints may not be detected or included in the candidate keypoints, or some noise keypoints are located in the candidate local zone, in order to get optimal matching path and to tolerate missing or noisy keypoints, we look backward more than one column in MS, namely, we consider more than one previous keypoint, which is already marked as a positive matching.

For a cell (c, j) in MS under consideration, only in two conditions we will update the values in $Hist(c, j)$ and $Score(c, j)$, based on $Hist(c', j')$ and $Score(c', j')$ under checking: one is the length of the matching path is longer, the other one is the matching score is smaller. Therefore, when the Algorithm 1 terminates, we find the longest matching path in $Hist$, and choose the one with minimum score in $Score$. So that we get a pair of $(score, path)$. In order to enhance the effect of the length of the matching path, the final matching score for a candidate local zone is $\frac{score}{size(path)^2}$.

1	3	2	10	9	11	15
	8	5	22	6	23	8
	16	7		4	7	4
		18		1		25

Fig. 7. An example of MS. The indexes in the jth column is the indexes of the keypoints in the query image, of which $LZ[j]$ is the candiditate keypoints.

D. Removing Overlapping Returned Results

Because we consider the candidate local zones for all the keypoints in the query image, many local zones are overlapped with each other. In order to remove the overlapping zones, all the local zones are first sorted by the length of the optimal matching path in the descending order and then sorted by their matching score in the ascending order. From the top one, we remove all the other zones with more than 50% overlapping areas, until we only have zones with no overlapping areas in the spotting list.

Algorithm 1 Calculate matching score

```
$w = max(size(M(kp'_j))), kp'_j \in LZ$
$MS = zeros(w, size(LZ))$
for $j = 1 \rightarrow size(LZ)$ do
    for $c = 1 \rightarrow size(M(LZ[j]))$ do
        $MS(c, j) = M(LZ[j])[c]$
    end for
end for
% Initialization
for $c = 1 \rightarrow size(M(LZ[1]))$ do
    $Hist(c, 1) = [MS(c, 1)]$
    $Score(c, 1) = d(MS(c, j), LZ[1])$
end for

for $j = 2 \rightarrow size(LZ)$ do
    for $c = 1 \rightarrow size(M(LZ[j]))$ do
        for each $MS(c', j'), j' \in [max(1, j-5), j-1]$ do
            if $MS(c', j') > MS(c, j)$ or $| Hist(c', j') | < | Hist(c, j) |$ then
                $Continue$
            end if
            if $MS(c', j') == MS(c, j)$ and $| Hist(c', j') | > | Hist(c, j) |$ then
                $Hist(c, j) = Hist(c', j')$
                $Score(c, j) = Score(c', j')$
            else
                if $| Hist(i', j') | +1 > | Hist(i, j) |$ or $Score(c', j') + d(MS(c, j), LZ[j]) < Score(c, j)$ then
                    $H(i, j) = H(i', j') \cup MS(i, j)$
                    $Score(c, j) = Score(c', j') + d(MS(c, j), LZ[j])$
                end if
            end if
        end for
    end for
end for
```

Furthermore, we remove the ones with lengths of matching paths smaller that $0.5 * n_q$. For the remaining zones, we normalize their matching scores by scaling to [0,1], and discard the ones with larger matching scores. For example, as shown in Fig. 8, we can see that after the 8th returned zone, the matching score increases dramatically, therefore, we only keep the first 8 zones.

IV. Experiments

A. Experimental Setup

We test our proposed method on Bangla handwritten documents, with two examples shown in Fig. 9. We select one document of each writer and segment it into word images, all of which will be used as query images to be spotted on all the other documents for the same writer in the experiments. For each query word, it will appear at least 8 times in every document. The word spotting results are evaluated by the average precision and recall.

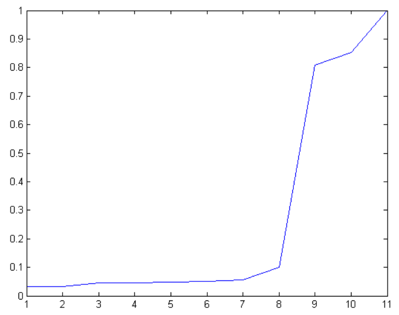
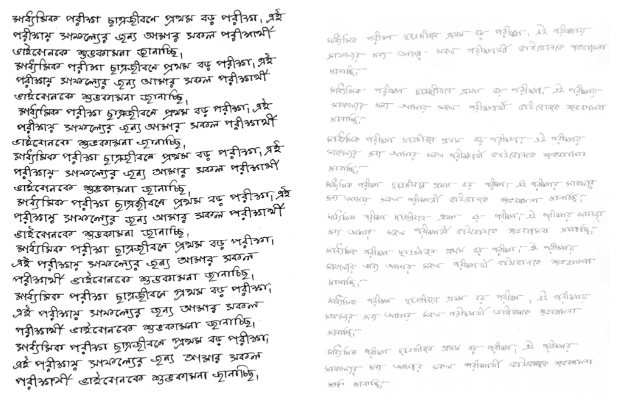


Fig. 8. Plot of the scores of returning zones. The horizontal axis is the index of the zones in the spotting list, and the vertical axis is the noramlzied matching score.



(a) (b)

Fig. 9. Two examples of Bangla handwritten documents.

For comparison, we will use SIFT features for the keypoints and apply our proposed matching method on the same set of query word images and documents.

B. Results

The experiment results are shown in Table I. Our method has both higher precision and recall than using SIFT features. HKS can tolerate variations better than SIFT, even the locations of the corresponding keypoints in the same occurring words are slightly different. Moreover, our method for deciding the size of the local patches can avoid bringing in noise into the patches, and includes as much information as possible. On the other hand, for SIFT, some keypoints have very large scales, so that the SIFT features are extracted from a large local region and always contain the parts of the surrounding characters. In this situation, these keypoints will not be matched correctly.

TABLE I. Experiment Results

Methods	Average Precision	Average Recall
SIFT with our proposed matching method	70.9	84.8
Our proposed method	77.2	94.8

In Fig. 10(c) and 10(d), there are two spotting results for the query images in Fig. 10(a) and 10(b). We can see that all the instances of the query images are spotted based on our method, but in Fig. 10(c), 10 spotted zones are returned, two of which contain only one character in the query image, and in Fig. 10(d), much more spotted zones are returned, because most of the parts in the query image appear in different words, even though only 8 instances are exactly the same to the query image, but all the other spotted zones have no more than one character different. For other longer query word images, with few common parts in the other words, the spotted zones in the spotting list are exactly the instances of the query images, as shown in Fig. 11.

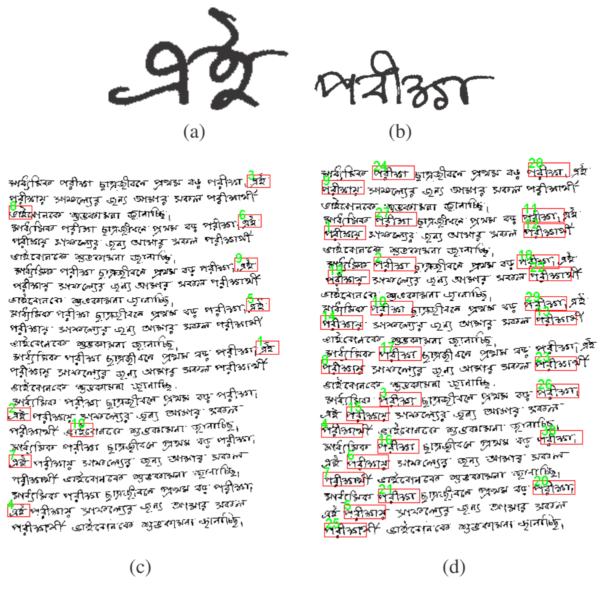


Fig. 10. (c) is the spotting results of (a). (d) is the spotting results of (b). The number marked around the spotted rectangle box is the position in the spotting list, namely, the smaller the number is, the more similar to the query image.

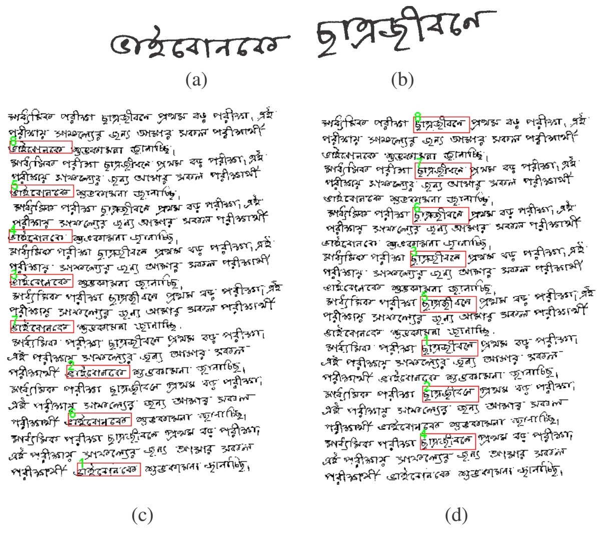


Fig. 11. (c) is the spotting results of (a). (d) is the spotting results of (b).

V. Conclusion

We propose a segmentation-free keyword spotting method for Bangla Handwritten documents. Keypoints detected both by the SIFT detector and morphological operations are used in order to avoid missing keypoints. The size of the patch centered at a keypoint is automatically determined by the entropy, rather than fixing the size of all patches to the same. In the searching process, in order to narrow down the searching scope, we only consider the local zones containing similar keypoints with respect to the ones in the query image, and the zones with enough matching keypoints are returned, sorted by the matching scores. According to our experiment results, HKS can tolerate the variations better than SIFT features, and our proposed method have higher precision and recall.

In future, we will research on reducing the dimension of HKS descriptors, and speed up the searching process. Then, we would like to work on the historical Indian documents, with degradations, multi-languages, multi-scales and rotations.

References

[1] R. Manmatha, C. Han, and E.M. Riseman, "Word spotting: A new approach to indexing handwriting," *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pp. 631–637, 1996.

[2] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Hmm-based word spotting in handwritten documents using subword models," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 3416–3419.

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855–868, 2009.

[4] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 211–224, 2012.

[5] Yann Leydier, Asma Ouji, Frank LeBourgeois, and Hubert Emptoz, "Towards an omnilingual word retrieval system for ancient manuscripts," *Pattern Recognition*, vol. 42, no. 9, pp. 2089–2105, 2009.

[6] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós, "Browsing heterogeneous document collections by a segmentation-free word spotting method," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 63–67.

[7] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] F. Moreno-Noguer, "Deformation and illumination invariant feature point descriptor," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1593–1600.

[9] Xi Zhang and Chew Lim Tan, "Segmentation-free keyword spotting for handwritten documents based on heat kernel signature," in *Document Analysis and Recognition (ICDAR), 2013 International Conference on*. IEEE, 2013.

[10] Harris Chris and Stephens Mike, "A combined corner and edge detector.," in *Alvey vision conference*, 1988, pp. 147–151.

[11] Tony Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[12] T Yung Kong and Azriel Rosenfeld, *Topological algorithms for digital image processing*, Access Online via Elsevier, 1996.

[13] Louisa Lam, Seong-Whan Lee, and Ching Y Suen, "Thinning methodologies-a comprehensive survey," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 9, pp. 869–885, 1992.