Performance Improvement in Local Feature Based Camera-Captured Character Recognition

Takahiro Matsuda, Masakazu Iwamura and Koichi Kise
Dept. of CSIS, Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka, Sakai, Osaka, 599-8531 Japan
matsuda@m.cs.osakafu-u.ac.jp, {masa, kise}@cs.osakafu-u.ac.jp

***Abstract*—Concerning camera-captured Japanese character recognition, we have proposed a method to recognize characters, both simple and complex, that may not be linearly aligned and may be printed with a complex background. Recognition is performed based on local features and their arrangement. The arrangement is validated with an algorithm called local RANSAC. However, at least four corresponding local features are required. To relax that condition, we propose a new recognition method making it possible to recognize a character region with at least three corresponding local features. This method enables recall and precision to be improved with the simpler characters using more corresponding local features and computation times to be reduced by 7%.**

***Keywords*—scene character recognition; local feature; RANSAC; reference point; affine transformation matrix;**

I. Introduction

Recognizing characters in a scene is a challenging and unsolved problem. To pursue better performance, much effort including competitions (e.g., [1]–[3]) and research (e.g., [4]–[12]) has been spent. With some exceptions (e.g., [3], [8]–[12]), most research is focused on Latin characters. Characters in other languages are usually more complex and have a different set of difficulties.

We address the problem of recognizing Japanese characters, including complex characters such as Kanji (Chinese characters), that may not be linearly aligned and may be printed with complex backgrounds. The solution proposed is based on local features (e.g., SIFT [13]) and their arrangement [11]. The idea is that if the arrangement of local features in a region of the query image corresponds to the arrangement of a reference character image, it is probable that the reference character exists in that region. The correspondence is validated using an affine transformation matrix (ATM) that projects the region of the query image onto a reference image. In [11], the ATM is estimated using the local RANSAC method proposed in the paper. Three or more corresponding local features are required for the estimation and at least one corresponding local feature is required to validate the estimated ATM. Hence, in total, at least four are required for this strategy. However, for simpler characters, such as Hiragana and Katakana, only a limited number, sometimes less than four, of local features are obtained, thereby preventing these characters to be recognized.

To relax the condition, we propose an alternative method making it possible to recognize a character region with at least three corresponding local features. Specifically, a reference point (RP) [14] is introduced to estimate the center of a character region in the query image and four parameters of the ATM in a step-by-step manner. Our method also enables recall and precision of the simpler characters to be improved using more corresponding local features; precision is thereby enhanced by about 20% with the same recall. As a consequence, the method also reduces computation times by 7% on average.

II. Related Work

Existing scene character recognition methods are classified into three approaches, termed texture based (sliding window based), connected components based, and local feature based. Although the first two approaches are actively researched, the third has received little attention.

Compared with the methods used in the other two approaches, executing the methods used in the local feature based approach takes time, especially in feature extraction, even if the features are good. Nevertheless, they work on a severe environment, such as in complex layouts and backgrounds. To compensate for the computation time spent in the process, we introduced the notion of the anytime algorithm into scene character recognition [12]; the idea is that among characters in the query image, easily recognized characters are outputted earlier and the more difficult ones later. This is achieved by extracting local features in a step-by-step manner. Whereas the local RANSAC was used in [12], it is replaced with the proposed method to exploit advantages that we shall discuss below.

In creating a large dataset, we had already introduced the RP in scene character recognition in the task of automatic labeling [15]. In that work, however, there was no need to take the number of correspondences into account because a sufficient number of densely sampled local features were available. In addition, instead of an affine transformation, a similarity transformation was assumed. Thus, the contents in the current paper is not trivial but sufficiently novel.

III. Recognition Method Using Local RANSAC

In this section, we review the conventional method using local RANSAC [11]. Figure 1 shows an overview of the strategy. It is based on local features and their arrangement, which are often used in object recognition. The idea is that if local features are located in the query image in the same arrangement as those in a reference image, the character of the reference image should be found within the region of the query image. The arrangement of the features is confirmed using the local RANSAC algorithm, which is a variant of the

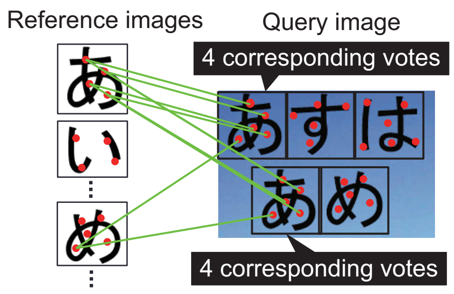


Figure 1: Overview of the conventional method [11]. Red points represent local features extracted and green lines indicate correspondences of features to reference images. Recognition results (characters and their bounding boxes) are determined at the same time based on correspondences of local features and their arrangement. Many correspondences are omitted for better clarity.

RANSAC algorithm [16], where the RANSAC algorithm is applied to a local region.

The recognition process consists of three steps: feature extraction, feature matching, and character detection and recognition. We explain each step in greater detail.

A. *Feature extraction and matching*

In feature extraction, we employ the SIFT algorithm [13] to detect and describe local features. SIFT features are known to be similarity invariant and robust to affine transformation.

In the feature matching step, each feature vector in the query image (hereafter called the *query feature vector* or *query feature*) is paired with one in the database that is the closest match in the feature space. For a fast look-up of the corresponding feature vectors, we employed bucket-distance hashing, a state-of-the-art approximate-nearest neighbor search method [17].

B. *Character detection and recognition*

Character detection and recognition is processed as follows. First, a reference image is selected. Only the query feature vectors paired with the feature vectors of the selected reference image are prepared. Second, a query feature vector is selected. Then, the local RANSAC algorithm is applied to a local region which is the surrounding region (a circular region of radius 100 pixels is used in this paper) of the selected feature vector. This procedure is carried out for all query feature vectors and all reference images.

We briefly describe the recognition strategy with the local RANSAC algorithm. In the first step, an ATM (hypothesis) is estimated using three correspondences randomly selected in the local region. In the second step, the estimated parameters are validated; a number e of correspondences are used to evaluate the estimated parameters. The evaluation criteria are n *hypothetical inliers*; a *hypothetical inlier* is a correspondence whose distance between a query feature projected to the reference image using the ATM and the corresponding feature in the reference image is smaller than a predetermined threshold t. The process is repeated r times, and the best ATM that realizes the largest number for n is selected. Increasing r improves the recognition accuracy. A character region that is estimated using the ATM and passes the evaluation is regarded as a character candidate.

Finally, each character candidate is evaluated based on a score that yields a confidence value. The score is calculated by

$$\text{score} = 100\frac{m_p}{\sqrt{N_p}}, \tag{1}$$

where m_p is the number of corresponding feature points inside the character region, and N_p is the number of feature points in the corresponding reference image. $\sqrt{N_p}$ is used to normalize the difference between the number of feature points extracted from each reference character. If the score is larger than a predetermined threshold T, the character candidate is outputted. In the event that several characters are outputted in almost the same region, we group the characters and the one with the highest score is chosen as the recognition result.

IV. Proposed Method

In the character detection and recognition step of the conventional method, the local RANSAC algorithm is used. We describe its replacement next.

A. *Affine transformation matrix*

An affine transformation maps a point (x, y) in a plane onto another point (x', y') in the same plane, given by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \boldsymbol{M}(a,b,c,d)\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \tag{2}$$

where

$$\boldsymbol{M}(a,b,c,d) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{3}$$

is a 2×2 matrix representing four independent transformations, and $\begin{pmatrix} e \\ f \end{pmatrix}$ is a column vector representing a translation. The four independent transformations are scalings

$$\boldsymbol{L}(\beta) = \begin{pmatrix} \beta & 0 \\ 0 & \beta \end{pmatrix}, \tag{4}$$

rotations

$$\boldsymbol{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{5}$$

independent scalings

$$\boldsymbol{A}(\alpha) = \begin{pmatrix} \alpha & 0 \\ 0 & 1/\alpha \end{pmatrix} \tag{6}$$

and shear

$$\boldsymbol{S}(\phi) = \begin{pmatrix} 1 & \tan\phi \\ 0 & 1 \end{pmatrix}. \tag{7}$$

In general, $\boldsymbol{M}(a,b,c,d)$ can be decomposed as matrix product

$$\boldsymbol{M}(a,b,c,d) = \boldsymbol{S}(\phi)\boldsymbol{A}(\alpha)\boldsymbol{R}(\theta)\boldsymbol{L}(\beta). \tag{8}$$

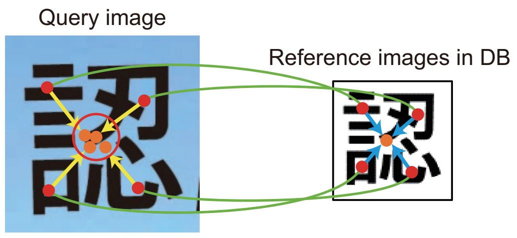


Figure 2: Determining the reference point (RP).

As an affine transformation is described by six parameters, it is determined from at least three corresponding points. We present two methods to estimate the translation and four independent transformations from three or more corresponding points. To avoid adverse effects from outliers in both methods, we use a robust estimator *median* instead of the mean.

B. Reference Point

We present a method to estimate the translation vector of the affine map. Figure 2 illustrates how the RP is determined. The RP (orange circle) is centered with each reference image. In the training phase, relative positions (blue arrows) from the feature points (red circles) to the RP are calculated for each reference image. In the recognition phase, the features in a query image are paired with those of a reference image. The relative positions to the RP are used to estimate the RP in the query image. If the features are correctly extracted and matched, the estimated RPs are clustered. We define R as the radius of the cluster. In such cases, the medians of the x- and y-coordinates of the estimated RPs in the query image are used as the estimate of the center. The RPs laying outside the clusters are discarded because they are regarded as having low reliability.

C. Estimation of the four independent transformations

Having determined the center of a character region in the query image, we need at least two correspondences to estimate the four independent transformations. However, it is accepted that more correspondences help obtain a better estimate. Hence, we present a method of estimating the transformations using not just two but more corresponding points. We use the decomposition given in Eq. (8) and estimate the parameters in the following way.

Scale: Scale β is estimated from the ratio of two corresponding distances between the center and a point. We take the median of the ratios.

Rotation: Rotation angle θ is estimated from the differences in two corresponding angles between the x-axis and a point assuming the center as origin. The median of the differences gives the rotation angle. However, taking the median of rotation angle is problematic because no boundary exists in the angle space. Hence, we employ a two-stage approach. First, to determine the boundary, we take the mean in the vector space; each angle is represented by a vector (i.e., $(\cos\theta_i, \sin\theta_i)$); therefore the mean angle is calculated using $\arctan\left((\sum_i \sin\theta_i)/(\sum_i \cos\theta_i)\right)$. Then, we can safely determine the boundary; for example, if the mean angle is 60 deg., the boundary is set at 240 deg. (-120 deg. on the other side). Finally, expanding the angle space, we take the median.

Independent scaling: The parameter α of independent scaling is estimated similarly to scaling. The difference is that the distances are calculated horizontally and vertically separately; the ratio of the horizontal and vertical distances gives α.

Shear: Shear maps a point (x, y) onto another point (x', y') given by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \boldsymbol{S}(\phi) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + y\tan\phi \\ y \end{pmatrix}. \tag{9}$$

Then, taking the difference between initial and final vectors, we can obtain

$$\begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} y\tan\phi \\ 0 \end{pmatrix}. \tag{10}$$

Hence, $(x' - x)/y$ gives an estimate of $\tan\phi$. We take the median of the value.

The four transformations are estimated and rectified one by one in the order specified in Eq. (8). However, there is an exception; scaling and rotation are independent of each other (as their matrices commute) and hence can be estimated simultaneously.

V. Experiments

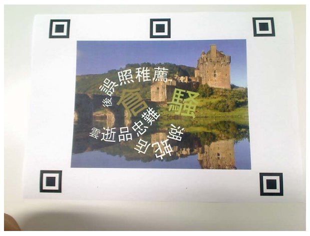
Before performing experiments, we have to prepare the query images. In this experiment, we used character sets in MS Gothic font. Because there is no appropriate dataset, we created one for our study. As the cost of collecting a large number of labeled data is expensive, we attempted to create a dataset semi-automatically. First we prepared source images with different characters and backgrounds; class labels and exact positions of the characters are ground-truthed. We printed it out and took pictures in various orientations. We then calculated a perspective transformation matrix between the source image and the pictures using five points estimated from the positions of the symbols used in the QR code, as shown in Figure 3. Telling more detail, although using just four symbols is sufficient to calculate the matrix, the other symbol is used to identify the corresponding symbols. By using the symbols, we can correctly calculate the transformation matrix because it is easy to recognize the symbols and obtain the locations. The symbols are used only to create ground truths and not used for character recognition.

In the experiments, we employed as reference data 2458 images of Hiragana, Katakana and Kanji characters in MS Gothic font. We prepared 120 query images composed of two types of background, and two sets of characters from the three scripts. Figure 3 shows some examples of queries. The resolution of the camera was 1600×1200. All experiments were performed on a computer with Opteron 2.60GHz CPU and 512GB memory. Each program was executed as a single thread on a single core. Hereafter, the conventional method and the proposed method are referred to as RANSAC and RP, respectively.

Figure 4 shows some recognition results obtained using the proposed method with the best parameters shown in Table I. As already discussed in [11], [12], recognition accuracy depended

(a) Hiragana (1-1)

(b) Katakana (4-3)


(c) Kanji (5-5)

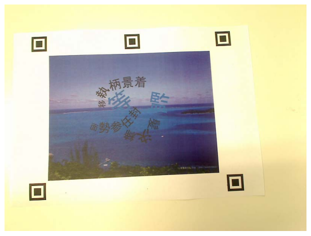
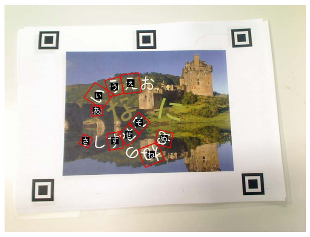
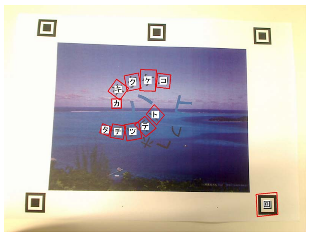
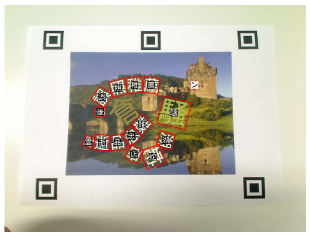

(d) Kanji (8-7)

Figure 3: Some examples of query images. Fifteen characters were placed on each image. We prepared two different sets of characters and two different backgrounds for each character type. The template ID (left) and photo ID (right) are given in parentheses after the character type.


(a) Hiragana (1-1)
Recall=0.667, Precision=1.000


(b) Katakana (4-2)
Recall=0.667, Precision=0.909


(c) Kanji (5-3)
Recall=0.933, Precision=0.933

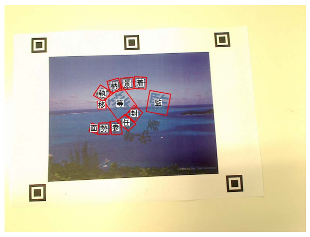

(d) Kanji (8-4)
Recall=0.800, Precision=1.000

Figure 4: Some recognition results obtained using the proposed method. A red frame encloses the character region of each recognized character and the recognition result is centered within the frame.

TABLE I: Best parameters used in the experiments of Figures 4 and 5. These parameters were experimentally found using the highest F-measure criterion.

	N	T	t	r	R
Local RANSAC (Conventional method)	5	75	3	50	-
Reference Point (Proposed method)	3		10	-	10

TABLE II: Recall and precision when $T = 0$ was used and N was changed in the proposed method (RP).

	Recall			Precision		
N	Hiragana	Katakana	kanji	Hiragana	Katakana	Kanji
3	0.56	0.53	0.88	0.49	0.41	0.59
4	0.51	0.45	0.86	0.65	0.54	0.72

on the number of features obtained from characters in query images; this is related to the complexity of character shapes. The trend was that less features were obtained from simple characters and more features were obtained from a complex character. The complex shape of Kanji characters in the query image contributed to high recall/precision because many local features can be extracted from them.

Figure 5 shows the relationship between recall and precision in RANSAC and RP. The same parameters were used in Figure 4 except that t was changed from 2 to 30. With threshold values given by $N = n + 3$, from 3 to 7 were used in RP, and from 4 to 7 were used in RANSAC. The figure shows that the precision for Hiragana and Katakana was higher in RP than in RANSAC. Figures 5(a) through 5(d) show the relationships between recall and precision for Hiragana and Katakana characters. For Hiragana, with recall of 0.45, precision was improved from 0.75 to 0.96; for Katakana, with recall of 0.35, precision was improved from 0.63 to 0.88. Comparing with RANSAC and RP for the same parameters, RP obtained better results in the F-measure criterion. That is to say, an ATM estimated using RP exhibited better performance than that using RANSAC. Figures 5(e) and 5(f) show the relationships between recall and precision for Kanji. For Kanji, there is no big difference between the two methods. The inference is that many features can be extracted from Kanji characters thereby improving the estimates of the ATM with RANSAC; the probability of obtaining a good ATM is high if there are plenty of available features to calculate the ATM with RANSAC. Overall, these results indicate that with better estimates of ATMs our proposed method performs character recognition better in comparison with the conventional method.

Table II shows the recall and precision when $T = 0$. A low value of T means that those recognized characters with low reliability are not filtered out. This can lead to an increase in character misrecognition. In this case, we found that RP with $N = 3$ does improve recall compared with $N = 4$ over all scripts. However, there was a side effect in that precision decreased significantly. Hence, we used a value $T = 75$ in the above experiments.

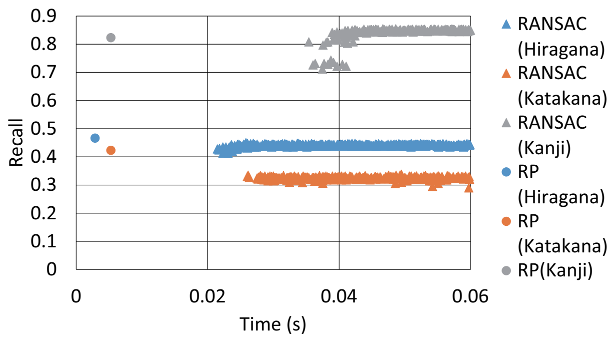


Figure 6: Relationship between processing time and recall.

TABLE III: Computation time (s).

	Feature extraction	Feature matching	Character detection and recognition	Total
Local RANSAC (Conventional method)	2.06	0.031	0.15	2.24
Reference Point (Proposed method)			0.004	2.10

Figure 6 shows the relationship between computation time and recall in character detection and recognition. We varied r from 1 to 50. The figure shows that RANSAC took more computation time than RP to achieve results in the same recall for all scripts. This is because RP is based on clusters; there is no need to explore query features to estimate good parameters.

Table III shows average computation times for both conventional and proposed methods in each process as well as the total. In particular, the computation time for character detection and recognition was reduced from 0.15s to 0.004s. The average total computation time was reduced from 2.24s to 2.10s representing a reduction of 7% in computation time.

VI. Conclusion

We focused on improving recognition performance in the task of camera-captured Japanese character recognition. We tackled the problem of recognizing both simple and complex Japanese characters that may not be linearly aligned line and may be printed with complex backgrounds. We proposed a method using local features and their arrangement. The arrangement was validated with the local RANSAC algorithm. However, this method requires at least four corresponding local features. To relax the condition, we proposed a new recognition method making it possible to recognize a character region with at least three corresponding local features. This method also helped in improving recall and precision of simpler characters using more corresponding local features and in reducing computation times.

Experimental results showed that we succeeded in improving precision of Hiragana and Katakana characters; for Hiragana, when recall was 0.45, precision was improved from 0.75 to 0.96; for Katakana, when recall was 0.35, precision was improved from 0.63 to 0.88. In addition, computation times in character detection and recognition was reduced from 0.15s to 0.004s, and the average computation time in total was reduced 7% from 2.24s to 2.10s.

For the future, we plan to unify the proposed method with the anytime algorithm [12] and automatic database creation [15]. The proposed method is more effective when many local features are used and would be suited for example in performing dense sampling of local features or in application to automatic database creation.

Acknowledgment

This work was supported by JST CREST project.

References

[1] A. Shahab, F. Shafait, and A. Dengel, "ICDAR 2011 robust reading competition challenge 2: Reading text in scene images," in *Proc. ICDAR*, 2011.

[2] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras, "ICDAR 2013 robust reading competition," in *Proc. ICDAR*, 2013.

[3] D. Kumar, M. A. Prasad, and A. Ramakrishnan, "Multi-script robust reading competition in ICDAR 2013," in *Proc. MOCR*, 2013.

[4] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang, "Scene text recognition using part-based tree-structured character detection," in *Proc. CVPR*, 2013.

[5] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky, "Large-lexicon attribute-consistent text recognition in natural images," in *Proc. ECCV*, 2012.

[6] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar, "Whole is greater than sum of parts: Recognizing scene text words," in *Proc. ICDAR*, 2013, pp. 398–402.

[7] L. Neumann and J. Matas, "On combining multiple segmentations in scene text recognition," in *Proc. ICDAR*, 2013.

[8] S. Lee, M. S. Cho, K. Jungz, and J. H. Kim, "Scene text extraction with edge constraint and text collinearity," in *Proc. ICPR*, 2010.

[9] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *Proc. VISAPP*, 2009.

[10] T. Yamazoe, M. Etoh, T. Yoshimura, and K. Tsujino, "Hypothesis preservation approach to scene text recognition with weighted finite-state transducer," in *Proc. ICDAR2011*, 2011.

[11] M. Iwamura, T. Kobayashi, and K. Kise, "Recognition of multiple characters in a scene image using arrangement of local features," in *Proc. ICDAR*, 2011.

[12] T. Kobayashi, M. Iwamura, T. Matsuda, and K. Kise, "An anytime algorithm for camera-based character recognition," in *Proc. ICDAR*, 2013.

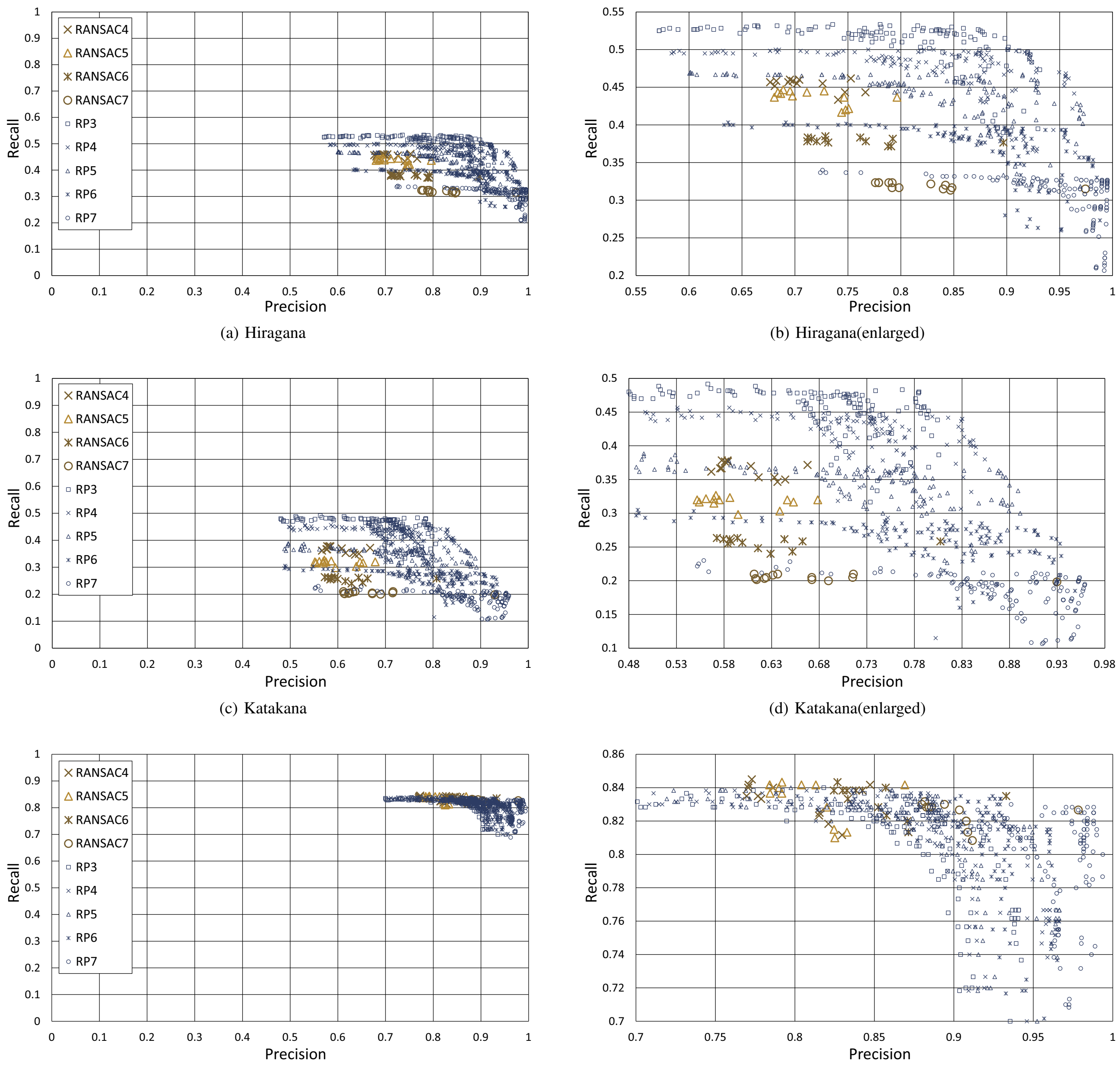
[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[14] M. Klinkigt and K. Kise, "Using a reference point for local configuration of SIFT-like features for object recognition with serious background clutter," *IPSJ Transactions on Computer Vision and Applications (CVA)*, vol. 3, pp. 110–121, 2011.

[15] M. Iwamura, M. Tsukada, and K. Kise, "Automatic labeling for scene text database," in *Proc. ICDAR*, 2013.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] M. Iwamura, T. Sato, and K. Kise, "What is the most efficient way to select nearest neighbor candidates for fast approximate nearest neighbor search?" in *Proc. ICCV*, 2013.



(a) Hiragana

(b) Hiragana(enlarged)

(c) Katakana

(d) Katakana(enlarged)

(e) Kanji

(f) Kanji(enlarged)

Figure 5: Relationship between precision and recall of the conventional method (RANSAC) and the proposed method (RP). The labels in the key legend of the scatter plots give the method and the value of the threshold N; for example, “RP3” means the RP method was used with $N = 3$. The images on the right side are enlarged ones of the corresponding ones on the left side.