Real-Time Recognition of Camera-Captured Characters in Complex Layouts

Masakazu Iwamura Tomohiko Tsuji Akira Horimtsu Koichi Kise
Dept. of Computer Science and Intelligent Systems, Osaka Pref. Univ., Japan
{masa,kise}@cs.osakafu-u.ac.jp, tsuji@m.cs.osakafu-u.ac.jp

Abstract

In order to realize an application of camera-based character recognition, a character recognition technique which is (1) ready for real-time processing, (2) robust to perspective distortion, and (3) free from layout constraints, is required. In this paper, a camera-based character recognition method which satisfy the three requirements is proposed. As a result of a combination of simple but efficient techniques, the proposed method enables us to execute camera-based character recognition in real-time even on a laptop PC with a cheap web camera. Real-time processing is realized using a principle of geometric invariants in a different manner than usual. This drastically reduced computational cost and achieved fast recognition of around 200 to 250 characters per second on a server.

1. Introduction

Camera-based character recognition has received considerable attention due to a wide variety of possible applications. One of convincing applications is "translation camera" which is a portable translating device integrated with a camera and a camera-based OCR [7]. Another possible one is to find useful words in the scene and tell them to blind people. The system is achievable by recognizing all characters captured by a camera and selecting registered words. In order to achieve the applications above, a character recognition technique which is (1) ready for real-time processing, (2) robust to perspective distortion, and (3) free from layout constraints, is required.

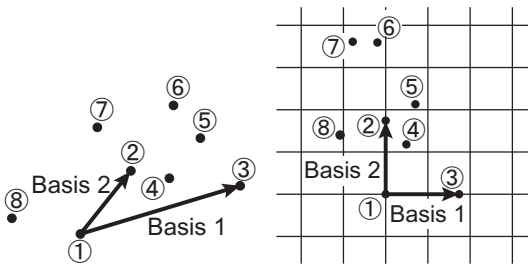
As a method satisfying the requirements (1) and (2), Myers proposed a method which extracts text lines from a scene image, eliminates geometric distortions of the texts and recognizes them [6]. However, the method cannot recognize a text shown in Fig. 1 and does not satisfy the requirement (3) because the text line is not straight and most characters are not upright. On the other hand, Kusachi et al. [4] and Li et al. [5] respectively proposed methods to recognize each deformed character. Although they satisfy the requirements (2) and (3), do not satisfy the requirement (1). Thus it has not been proposed any method satisfying the three requirements simultaneously.

Figure 1. An example of a designed text.

In this paper, we propose a simple but efficient implementation of character recognition that satisfies the three requirements simultaneously. The strategy of the proposed method is to recognize each deformed character as similar to Kusachi et al. [4] and Li et al. [5]. In order to realize real-time processing, the configuration of the proposed method is simple; adaptive binarization and contour extraction are used for segmentation and an improvement of geometric hashing (GH) like method whose computational cost is reduced from $O(P^4)$ to $O(P^2)$, where P is the number of feature points, is used for recognition. With help of new hashing and voting techniques, the proposed method runs well in real-time even on a laptop PC with a web camera.

1.1. Problem Definitions and Solutions

Let us confirm the problem definitions. First of all, we assume black characters are written on a white paper for simplicity. Since character images are captured by a camera, they can suffer from perspective distortion and be degraded by defocus and low resolution. We assume, however, connected components (CCs) of characters are extractable. We also assume all characters in the image exists



(a) Determination of bases.

(b) Projection to an affine invariant coordinate system.

Figure 2. An invariant coordinate system of geometric hashing.

on a flat paper (co-planer).

In the paper, we solve three problems: (i) fast recognition of extracted CCs, (Ii) Robustness enhancement of recognition, (iii) recognition of separated characters which consists of more than one CC such as "i" and "j." For (i), we describe a fast recognition method based on GH in Sec. 2. The method is improved using a principle of geometric invariant calculation. For (ii), a simultaneous estimation of class and pose of a character is presented in Sec. 4.2.4. For (iii), a method similar to generalized Hough transformation is given in Sec. 3.

2. Fast Recognition of Connected Components

2.1. Geometric Hashing

GH describes the image which has undergone a certain geometric transformation, by using invariant coordinate systems. In order to present the proposed recognition method, GH is presented in the affine transformation case.

2.1.1 Storage

Let us assume that we have feature points extracted from a reference image. First of all, three of them are rondomly chosen and two bases are defined as shown in Fig. 2(a). Then, as shown in Fig. 2(b), all the feature points are projected to the invariant coordinate system spanned by the bases. The invariant coordinate system is divided (quantized) into subregions in advance. Thus, the image ID and basis-set ID are stored into each corresponding subregion. The storage process finishes after the procedure above is carried out for all the invariant coordinate systems spanned by all the sets of bases and for all the reference images to be stored. The computational cost of storing an image is $O(P^4)$ where $O(P^3)$ is for creating an affine invariant coordinate system and $O(P)$ is for referencing the hash table.

2.1.2 Recognition

The initial phase of the process is almost the same as that of the storage process. Let us assume that we have feature points extracted from a query image. Three of them are chosen and two bases are defined (Fig. 2(a)) and all the feature points are projected to the invariant coordinate system spanned by the bases (Fig. 2(b)). The invariant coordinate system is divided into subregions and each of them corresponds to a bin of the hash table. Then, for each projected feature vector, a vote for the corresponding pairs of the image ID and the basis-set ID is cast. The procedure above is carried out for all the sets of bases. Finally, the pair of the image ID and the basis-set ID with the highest vote is determined. The process can quit when the output image is obvious. The computational cost of recognizing an image is the same as the storage process.

2.2. Improvement of Geometric Hashing

2.2.1 Difference in Problem Definition

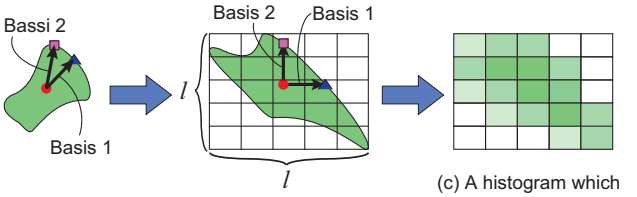
Before presenting the proposed method of recognizing CCs based on GH, let us present the difference in problem definition between GH and our problem.

The problem that GH solves is to identify the object only with the arrangement of feature points. This means that it does not take into account from what feature points are extracted. To the contrary, in our problem, we know feature points are extracted from a CC and the CC itself is also given. The additional information enables us to realize fast recognition.

Since we treat a recognition problem of a CC hereafter, we call the "image ID" "connected component ID (CC ID)." We use all pixels on the external contour of a CC as feature points in principle.

2.2.2 Use of Image Features

In GH, the arrangement of feature points are regarded as features for an image. To the contrary, the proposed method uses a feature vector representing the shape of a CC. The procedure of that is as follows. To begin with creating an invariant coordinate system as presented in Sec. 2.1. As the result, a figure of Fig. 3(a) is normalized into Fig. 3(b). Then, $k(= l \times l)$ uniform subregions are defined and a histogram of black pixels shown in Fig. 3(c) is created. Finally, the histogram is normalized to satisfy the sum of bins is 1 and a k-dimensional feature vector is obtained. As long as the same bases are selected, the vector is affine invariant in



(a) A figure (b) Normalization (c) A histogram which represents the figure (a)

Figure 3. Calculation of a k-dimensional feature vector based on values of k subregions. $k = l \times l$. For convenience' sake, black pixels are represented in green.

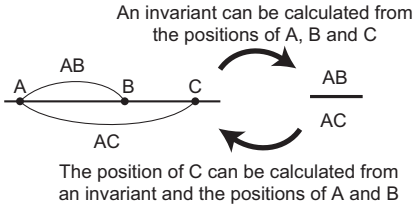
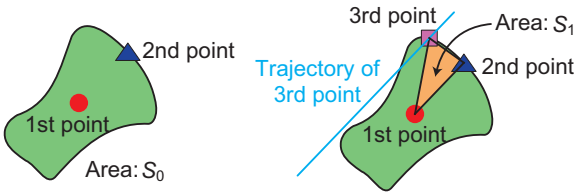


Figure 4. The relationship between the positions of points and an affine invariant.



(a) A figure with two points. (b) Determination of 3rd point.

Figure 5. Illustration of unique determination of the third point with area ratio.

theory. In this paper, we used $l = 4$ according to preliminary experiment results.

The reason why we use a feature vector calculated from the figure is because it is and can be more descriptive. Although the proposed method in the paper uses the histogram of black pixels, it is possible to use common features and normalization methods of characters used for scanner-based character recognition in the future.

2.2.3 Reduction in Computational Cost

The lack of GH is tremendous computational cost. The affine invariant version of GH requires $O(P^4)$ for recognition, where P is the number of feature points. This means that if we have 100 points, it requires $O(100,000,000)$! Thus GH is never applicable to real-time applications.

The reason why GH requires tremendous computational cost is that GH cannot determine the corresponding set of bases in the storage and recognition processes beforehand, though GH requires the correspondence for successful recognition [2]. This means that GH has to search the correspondence by examining all (or many) sets of bases. Thus if we can select the corresponding set of bases, the computational cost can be reduced. In the proposed method, we use a principle of geometric invariants in a different manner than usual.

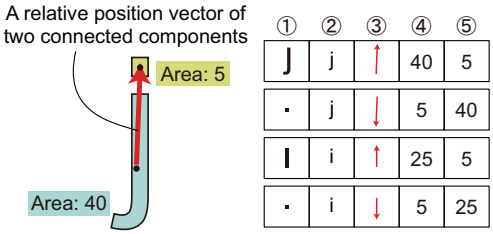
Before presenting the detail of the proposed method, we present the principle. As shown in Fig. 4, AB/AC, calculated from three points A, B and C, is an invariant to affine transformation. This is usually used. On the other hand, the position of C is determined by the value of the invariant and positions of A and B. Assuming A, B and C are on the same line, C can be the left side of A or the right side of B. If the order of the points are defined in advance, the position of C is uniquely determined. In the generalized form of the principle is that "If we have the value of the invariant and $n-1$ points out of n points, we can determine the last point almost uniquely under a geometric transformation." Such a unique determination of the position of a feature point helps to reduce computational cost.

The procedure of our proposed method is as follows. Firstly, we determine the first point uniquely using the nature that the center of gravity of a connected component does not change regardless of affine transformation [1]. Secondly, we have to select one point from the external contour randomly in the same manner as GH. Finally, we can determine the third point almost uniquely by our proposed manner of selecting a point described above. This procedure reduces the computational cost from $O(P^4)$ to $O(P^2)$. since the first and third points are determined uniquely.

However, the procedure above determines three points on the same line. This is actually useless to calculate linearly independent two bases required to create the invariant coordinate system. Thus we illustrate another method to determine the third point for a figure whose area is S_0 shown in Fig. 5(a).

Let us assume the third feature point is given as Fig. 5(b) and the area of the triangle determined by the three points is S_1. Then S_1/S_0 is an affine invariant. Therefore, the third point is determined so that S_1/S_0 equals a predetermined value or S_1/S_0 takes the maximum value. To determine uniquely the third point, the order in clockwise or anticlock-

[1] Although there is no guarantee that the center of gravity is on the external contour, this does not cause anything in the following processes.



(a) relative position vector (b) Separated character table

Figure 6. Description of separated character table. The elements of the table are, from left, ①Shape of the connected component (CC ID), ②original character, ③relational position of the paired CC, ④the area of the CC, ⑤the area of the paired CC.

wise is usable. As long as S_1 is a constant, the trajectory of the third point is a line parallel to the line through the first and second points as shown in Fig. 5(b). Therefore, the third point is easily determined as the intersection of the line and the external contour. If there are several intersections, it is also possible to determine uniquely the third point, for example, the furthest point is the one. In this paper, the third point is determined so that S_1/S_0 takes the maximum value.

3. Recognition of Separated Characters

In this section, we present a recognition method of separated characters which consists of more than one CC such as "i" and "j."

In order to do that, in the storage process, the number of CCs in a character is counted. If the number is greater or equals to two, each CC is stored in the database as if the CC were a character, and the character is registered into the separated character table shown in Fig. 6. The table stores the relative positions and sizes between the CCs of a character to recognize a separated character.

In the case of Arial font, the bottom CC of "i" has the same shape of "I (capital ai)" and "l (lowercase el)," and they are indistinguishable. Thus in order to recognize "i" correctly, each CC of the same shape such as "I" and "l" must be checked whether it is a part of "i" or not. Finally, if the top CC of "i" exists in the right position and size, the pairs of CCs are recognized as "i."

In order to realize the process above, all CCs in the same shape are stored so as to have the same CC ID. That is, storing reference character images is processed one by one and each image is checked whether CCs in the same shape are already registered or not. In further detail, before storing a reference image, it is recognized using the database in process of creation, and the same CC ID is assigned if CCs in the same shape are found. Ideally the bottom CC of "i," "I" and "l" have the same CC ID. However, the method turned out not to get along with the generative learning mentioned below, and some CCs did not have the same CC ID. Therefore, CC IDs were manually set up according to the similar character list shown in Table 1.

Table 1. List of similar characters. Characters in a cell were treated as the same class.

0 O o	6 9	C c	I l	S s	u n
W w	X x	N Z z	p d	q b	7 L V v

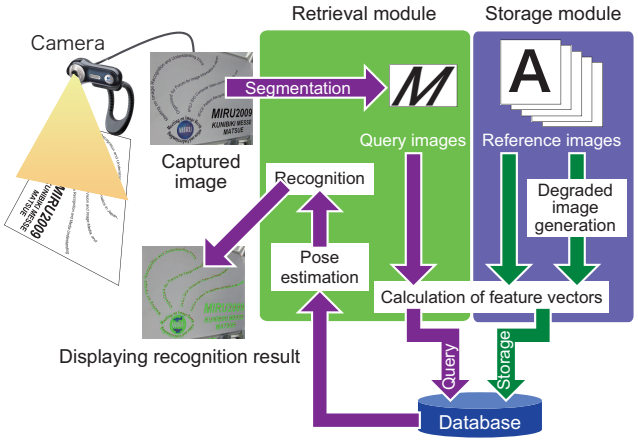


Figure 7. An overview of the proposed system.

4. Overview of the Proposed Method

The overview of the proposed system is shown in Fig. 7. The system consists of storage and retrieval modules.

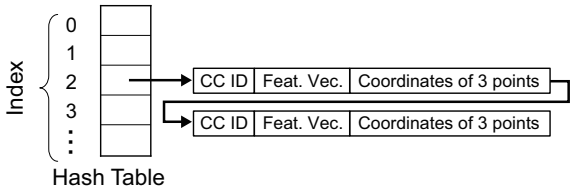


Figure 8. Configuration of the hash table.

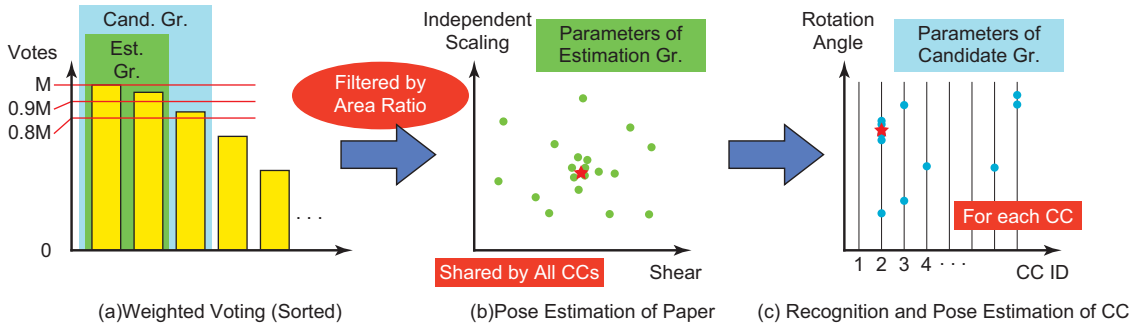


Figure 9. Recognition and pose estimation of a connected component.

4.1. Storage module

In the storage module, reference images are stored in the database. The reference images are binary images in the paper.

4.1.1 Degraded Image Generation

In order to cope with degradation caused by defocus and low resolution, degraded images are artificially created by generative learning [1]. In this paper, nine degraded images (including the original reference image) are created from each reference image by applying three kinds of Gaussian blurring (including no degradation) and three kinds of degradation in resolution (including no degradation). The created images are binalized and treated as reference images.

4.1.2 Feature Vector Calculation

Feature vectors are calculated in the manner presented in Sec. 2.2.2. They are concatenated in the following process. An invariant coordinate system shown in Fig. 3(b) is calculated from three feature points by regarding one point as the origin of the coordinate system. By replacing the point of the origin, three invariant coordinate systems are calculated. That is, from three points, three k-dimensional feature vectors are calculated. They are just concatenated and a $3k$-dimensional feature vector is obtained.

4.1.3 Storage into the Database

The proposed method uses the hash table shown in Fig. 8 to store the information of CCs. Each entry which consists of a CC ID, a feature vector and the coordinates of three points is stored using the list structure. The coordinates of three points is used for the pose estimation in the recognition process. Since the proposed method uses feature vectors, the hash table is a common 1-dimensional one instead of 2-dimensional one of GH.

The index H_{index} of the hash table is calculated as

$$H_{\text{index}} = \left(\sum_{i=1}^{3k} D^{i-1} r_i \right) \bmod H_{\text{size}}, \tag{1}$$

where H_{size} is the size of the hash table ($2^{19} - 1$ was used) and r_i is the quantized value of the i-th element into D levels. $D = 3$ was used according to preliminary experiment results.

4.2. Recognition module

4.2.1 Image Acquisition

An image to be recognized is captured by a digital camera or a web camera as a still image or a movie. A movie is decomposed into frame images. We call each obtained image "query image."

4.2.2 Segmentation

CCs are extracted from the query image. The image is adaptively thresholded into the binary image and a contour extraction technique is applied.

4.2.3 Feature Vector Calculation

Feature vectors are calculated from a CC. The process is almost the same as Sec. 4.1.2. The only difference is that the number of feature vectors is restricted to S for speeding up. The relationship between S and the recognition performance is discussed in Sec. 5.2.

4.2.4 Recognition and Pose Estimation

From the hash table shown in Fig. 8, CC IDs and the coordinates of three feature points are obtained using the feature vectors. The CC IDs are temporary recognition results and some of them are wrong. Thus we extract the correct results by use of a few steps of a voting procedure similar to [3].

That is, as shown in Fig. 9, pose estimation of the paper is done first and then recognition and pose estimation of each CC are done.

Firstly, from the correspondence between feature points in the query image and a reference image, the pose of the CC is calculated as an affine transformation matrix. Since some of them are wrong, they are filtered by weighted voting of CC ID for each CC as shown in Fig. 9(a). The reason why the voting is weighted is a CC which has longer external contour may have unfairly large number of votes. Letting N_i be the number of feature points (the length of the external contour) of i-th CC, the weight for the i-th CC is defined as $\frac{1}{\sqrt{N_i}}$. Let M be the number of the highest vote, and characters which have larger number of votes than $0.9M$ are grouped in "estimation group" and characters which have larger number of votes than $0.8M$ are grouped in "candidate group." These groups are defined for each CC of the query image.

Secondly, the pose of the paper is estimated. Since we assume all the characters exists on a paper, all CCs are expected to share the same parameters of shear and independent scaling. Thus, as similar to [3], a pair of plausible parameters are estimated by using density estimation in the 2D space as shown in Fig. 9(b). That is, affine transformation matrices of the estimation group are plotted in a 2D space and the densest point represented by a red star mark in Fig. 9(b) is selected. In order to increase reliability, only CCs satisfying $T_{\text{area}} \leq R/\beta^2 \leq 1/T_{\text{area}}$ are used for the estimation, where R is the area ratio of the CC of the query image and the corresponding CC of the reference image, and β is the scaling parameter calculated from the affine transformation matrix. $T_{\text{area}} = 0.7$ was used in the paper. Note that the value of R/β^2 far from 1 is incredible because if the temporary recognition result is correct, R/β^2 must be 1.

Thirdly, recognition result of each CC is determined. As shown in Fig. 9(c), a pair of plausible rotation angle and recognition result of the CC are estimated by using density estimation in the 2D space. Affine transformation matrices of the candidates group are used. The difference from Fig. 9(b) is that the density estimation is carried out in 1D space since the CC ID is discrete value. Finally, the process in this section estimates recognition result (CC ID) and pose (shear, independent scaling and rotation) of the CC.

5. Experiments

In order to confirm the effectiveness of the proposed method, two experiments were carried out on a server with Opteron 2.6GHz. In order to reduce the computational cost, reference images and query images were normalized so that the largest size of the width and height of an image was 100 pixels and 50 pixels, respectively.

(a) Arial, Century, Gigi and Impact from the top.

(b) Pictograms.

Figure 10. Samples of (a) fonts and (b) pictograms.

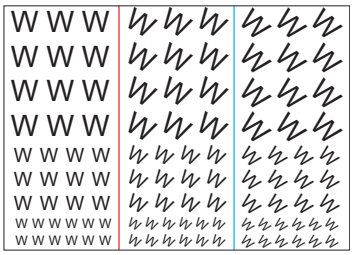


Figure 11. A sheet of recognition target.

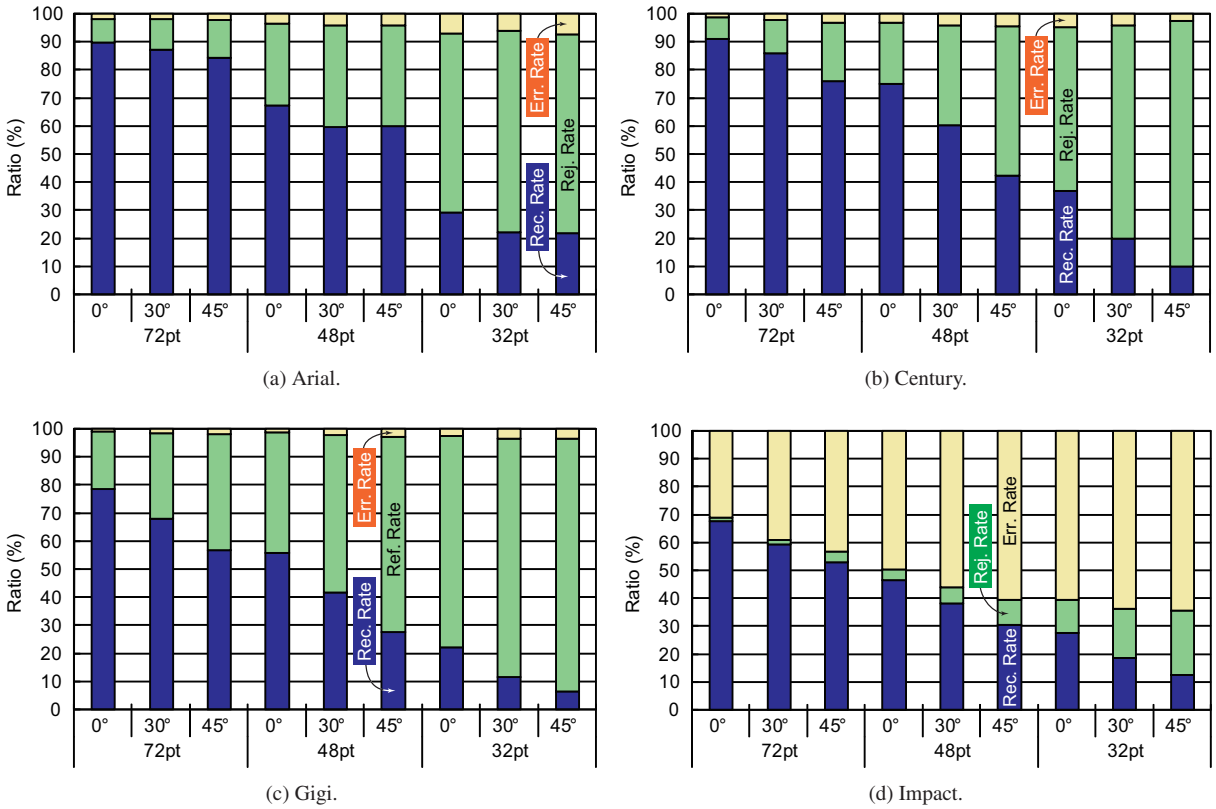
5.1. Fonts

We evaluated the effectiveness of the proposed method using characters in various fonts. We employed 62 characters of numerals and alphabets as shown in Fig. 10(a): 10 figures, 26 lowercase alphabets and 26 capital alphabets. As mentioned in Sec. 3, some characters are difficult to distinguish under affine distortions. Thus the characters in a cell in Table 1 were treated as the same class in all the experiments. In the recognition process described in Sec. 4.2.4, if the highest vote was 0 for a CC, the CC was rejected.

As recognition targets, we prepared 62 sheets of letters as shown in Fig. 11. Each of the sheet contained 108 characters evenly in nine conditions which were the combinations of three different sizes (72pt, 48pt and 32pt) and three different rotation angles (0, 30 and 45 deg.). Each sheet was captured in three different angles (0, 30 and 45 deg.) by a digital camera with a resolution of 1024×768. The average size of "A" of 72pt captured in 0 deg. was 40.7×44.8 pixels. That of "A" of 32pt captured in 45 deg. was 10.0×18.6 pixels. A parameter S presented in Sec. 4.2.3 was 20.

Firstly, the average processing time per character is shown in Table 2. Since the values were about 4ms, the recognition speed turned out to be around 200 to 250 characters per second by a simple calculation.

Secondly, the recognition results are shown in Fig. 12. The figures show that recognition rates decreased as the size of characters decreased or as the angle of capture increased.



(a) Arial.

(b) Century.

(c) Gigi.

(d) Impact.

Figure 12. Recognition rates of fonts in various sizes and angles of capture.

Table 2. Average processing time (ms) for recognition of a letter or a pictogram. Picto represents pictograms.

Font	Arial	Century	Gigi	Impact	Picto
proc. time	3.9	3.6	3.7	5.2	4.2

The figures also show that the decrease in recognition rate was caused by decrease in character size rather than change in angle of capture.

The detail of the recognition results are discussed below. (i) For the fonts other than Impact, as recognition rates decreased, rejection rates increased by that much and error rates did not increased so much. The cause was a large D. If D was large, the hash index changed by a slight fluctuation of an image. This decreased the number of reliable temporary results. When $D = 2$ was used instead of $D = 3$, the lowest recognition rate of Arial, in the case of 32pt and 45 deg., the recognition rate increased from 21.54% to 52.73% though the error rate also increased from 7.57% to 36.31%. It is up to applications which should be chosen. We chose $D = 3$ in this paper because we think lower error rates are desirable in general.

(ii) For Impact, as recognition rates decreased, error rates increased by that much and rejection rates did not increased so much. The cause was that feature vectors got similar and discrimination ability decreased. This seems to come from that characters in Impact have thick lines. The average processing time in Table 2 shows that Impact required more time. This implies that many hash collisions occur due to low discrimination ability of feature vectors. This can be improved by introducing features and normalization methods for scanner-based character recognition as mentioned in Sec. 2.2.2

Finally, in order to investigate recognition ability to figures, 10 pictograms shown in Fig. 10(b) were also recognized in the same manner of the fonts. As shown in Fig. 13 and Table 2, the recognition results were similar to the fonts other than Impact.

From the results above, we confirmed that the proposed method worked fast and achieved low error rates except a part of fonts.

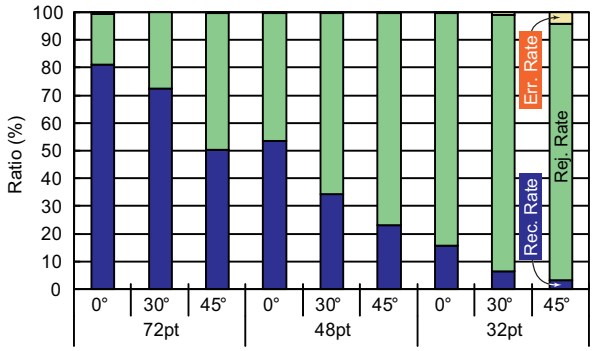


Figure 13. Recognition rates of pictograms in various sizes and angles of capture.

(a) 0 deg. (b) 30 deg. (c) 45 deg.

Figure 14. Camera-captured images of the designed text in Fig. 1 captured from various angles.

5.2. Designed text (Fig. 1)

The proposed method was applied to the designed text shown in Fig. 1. Camera-captured images of it from 0, 30 and 45 deg. are shown in Fig. 14. The sizes of the images were 2470×1746, 2278×1746 and 2038×1844, respectively. The figures contained 236 characters except commas; 168 in Arial, 27 in Arial Black Italic, and 41 in unknown fonts which existed around the blue logotype. The characters were recognized after storing Arial and the unknown fonts into the database.

Processing time and recognition results in $S = 200$ and $S = 20$ are shown in Table 3. Most errors were caused by misrecognition of "i" (confusion of "i," "I" and "l"), confusion of "U," "u" and "n," and that of "E" and "m." From the comparison of $S = 200$ and $S = 20$, $S = 20$ was about 6 times faster than $S = 200$. Thus if fast recognition is desired, smaller S is better. $S = 200$ achieved higher recognition rates than $S = 20$. Thus if higher recognition rate is desired, larger S is better.

Table 3. Recognition rates and whole processing time for designed text in Fig. 1.

S	200			20		
Angle (deg.)	0	30	45	0	30	45
Time(ms)	7990	7990	7020	1300	1260	1140
Rec. rate(%)	94.9	90.7	86.4	86.9	81.8	76.3
Rej. Rate(%)	0.4	3.0	6.4	6.4	9.3	16.5
Erro Rate(%)	4.7	6.4	7.2	6.8	8.9	7.2

6. Conclusions

In this paper, a camera-based character recognition method which satisfies (1) ready for real-time processing, (2) robust to perspective distortion, and (3) free from layout constraints, was proposed. The proposed method is executable in real-time even on a laptop PC with a cheap web camera though the result is not shown in the paper.

Future work includes introduction of features and normalization for scanner-based character recognition.

Acknowledgement This research was supported by KAKENHI 19700177 and 21700202.

References

[1] H. Ishida, S. Yanadume, T. Takahashi, I. Ide, Y. Mekada, and H. Murase. Recognition of low-resolution characters by a generative learning method. In *Proc. CBDAR2005*, pages 45–51, 2005.

[2] M. Iwamura, T. Nakai, and K. Kise. Improvement of retrieval speed and required amount of memory for geometric hashing by combining local invariants. In *Proc. BMVC2007*, volume 2, pages 1010–1019, Sept. 2007.

[3] M. Iwamura, R. Niwa, A. Horimatsu, K. Kise, S. Uchida, and S. Omachi. Layout-free dewarping of planar document images. In *Proc. DRR XVI*, 7247-36, Jan. 2009.

[4] Y. Kusachi, A. Suzuki, N. Ito, and K. Arakawa. Kanji recognition in scene images without detection of text fields—robust against variation of viewpoint, contrast, and background texture—. In *Proc. ICPR2004*, 2004.

[5] L. Li and C. L. Tan. Character recognition under severe perspective distortion. In *Proc. ICPR2008*, 2008.

[6] G. K. Myers, R. C. Bolles, Q.-T. Luong, J. A. Herson, and H. B. Aradhye. Rectification and recognition of text in 3-d scenes. *IJDAR*, 7(2-3):147–158, 2004.

[7] Y. Watanabe, Y. Okada, Y.-B. Kim, and T. Takeda. Translation camera. In *Proc. ICPR1998*, pages 613–617, 1998.