Robust and Efficient Recognition of Low-Quality Images by Cascaded Recognizers with Massive Local Features

Koichi Kise Kazuto Noguchi Masakazu Iwamura
Dept. of Computer Science and Intelligent Systems, Osaka Prefecture University
1-1 Gakuencho, Naka, Sakai, Osaka 599-8531, Japan
{kise, masa}@cs.osakafu-u.ac.jp

Abstract

For image recognition with camera phones, defocus and motion blur cause a serious drop of the image recognition rate. In this paper, we employ generative learning, i.e., generating blurred images and learning based on massive local features extracted from them, for a recognition method using approximate nearest neighbor search of local features. Major problems of generative learning are long processing time and a large amount of memory required for nearest neighbor search. The problems become serious when we use a large-scale database. In the proposed method, they are solved by cascaded recognizers and scalar quantization. From experimental results with up to one million images, we have confirmed that the proposed method improves the recognition rate, and cuts the processing time as compared to a method without generative learning.

1. Introduction

As mobile devices with cameras such as camera phones have become more common, it is natural to make use of them as input devices to access information on the Internet. Such applications can be thought of as a replacement of bar-codes, and include information retrieval by recognizing pictures of items in catalogs, book covers and CD jackets, etc. As compared to well-studied *generic* object recognition which is to find classes of objects, the above recognition is called *specific* (or *particular*) object recognition to distinguish instances of objects.

In order to make the above recognition services practical, recognition methods should satisfy the following requirements: (1) large-scale, (2) memory-efficient, (3) fast, (4) robust, and (5) near error-free. Large-scale recognition is necessary for services on a large number of objects, say 1 million. Large-scale recognition requires methods to be memory-efficient, because the amount of required memory on servers directly affects the cost of services. Fast recognition is also required for the same reason. Robust recognition is needed for low-quality images (query images), i.e., images captured with various imaging conditions including different lighting, occlusion, low-resolution, defocus and motion blur. The latter two factors are crucial for recognition with mobile devices. The final requirement, near error-free recognition is based on the observation that, from the users' point of view, it is far better to reject a query image than to receive an erroneous result.

So far, researchers have attempted to build specific object recognition. Pioneering work by Schmid and Mohr [8] as well as Lowe [5] have been extended to give methods to better fulfill some of the above requirements [9, 6, 7]. However, further investigation is required to meet the requirements at a practical level.

In this paper, we propose a method of recognition by matching local features aiming at satisfying the requirements as follows. For recognition of low-quality query images with a database of 1 million images, 90% of query images can be recognized correctly with the 32GB memory in 50ms/query. Rejection of 10% of query images allows us to recognize images with the error rate less than 1 %.

The most important contribution of this work is to discover the fact that the key to make robust recognition of low-quality images more *efficient* is to *increase* massively the number of local features for indexing images in the database. Such a paradoxical "fact" is realized by a new paradigm of cascaded recognizers with the two important properties *monotonicity* and *difference accessibility*.

2. Related work

In this paper, we focus on specific object recognition based on local features such as SIFT. Even for this limited field of recognition, many methods have so far been proposed. They can be characterized by the following four factors: representation, robustness, efficiency, and memory.

The representation is how to represent images based on local features. Basic methods are twofold: representation

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

using visual words and representation using a set of *raw* local features. Although visual word representation is dominant for generic object recognition, it causes a problem for specific object recognition. A considerably large number of local features need to be retained as visual words. In [6], it is said that, for better recognition, a visual word should represent only a few local features, which causes problems both on processing time and memory. We have also tested using our image samples and obtained similar results. The representation using a set of raw local feature vectors is to keep original vectors as representation. It was introduced at the beginning of specific object recognition as in [5] and still has its advantage thanks to its simplicity. The visual word representation is to describe an image with the vector space defined by visual words. Since the resultant vector is global, i.e., a single vector for a whole image, it cannot deal with occlusion and some other local changes in the image. The representation as a set of local features can solve such problems since local changes are limited to destroy some local features; the rest are still good for recognizing objects.

With the representation of a set of raw local features, matching feature vectors is a fundamental processing for recognizing objects. The simplest way is to find an object whose local features match best to those of a query image. The simplest process of matching is to find the most similar feature vector by nearest neighbor search.

Since the number of feature vectors are so large, it is required to apply speed-up technologies for matching. Lowe has proposed a method called Best-Bin-First algorithm [5] based on the k-d tree. This is to approximate the nearest neighbor search to obtain significant speed-up. Ke and Sukthankar employ a hash-based approach [3] for the approximation using LSH (Locality-Sensitive Hashing) [1].

In order to achieve a high recognition rate, methods for generic object recognition employ sophisticated classifiers and distance measures such as SVM (support vector machine) and EMD (earth mover's distance). However these technologies are difficult to apply to specific object recognition due either to its high dimensional representation or a large number of feature vectors. Another possible way is to generate degraded images and extract local features from them to supplement the originals. For example, the method with random ferns [7] is a successful case of this approach, which is sometimes called *generative learning* [2].

It can be explained using analogy of shooting. The approach to generic object recognition is to improve the preciseness of shooting. On the other hand, the approach to specific object recognition is for hitting a target even by a bad shot: to increase the number of either bullets or targets by generative learning.

Memory is another source of problems when a large scale recognition is needed. The key is how to compress local features that are typically represented as high-dimensional real-valued vectors. One way is to reduce the dimensionality. PCA-SIFT [3] is a variant of SIFT for this purpose. Another way is to quantize feature vectors. Vector quantization, which generally allows us a better compression rate, is employed for obtaining visual words (each visual word corresponds to a codeword by vector quantization). Scalar quantization is also used and found that, as compared to harmful vector quantization, scalar quantization affects almost nothing down to 2 bit/dim. representation [4].

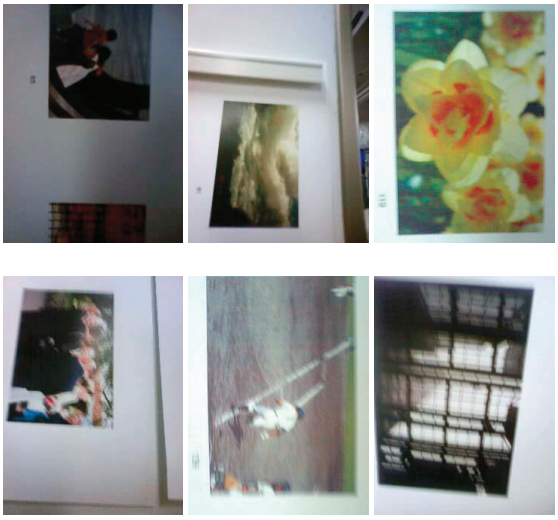


Figure 1. Examples of query images taken by camera phones.

3. Generative learning for low-quality images

3.1. Task

Figure 1 shows typical examples of images obtained with several camera phones by taking small printouts of original pictures. Various imaging problems including low resolution (QVGA), defocus, motion blur, uneven lighting, and perspective distortion make their recognition hard. In addition, some images are off the frame and sometimes include a different picture due to the time lag of the shutter.

The task of recognition in this paper is to recognize correctly such low-quality images in an efficient and robust way. It may not be difficult if the number of images to be distinguished is not large. However, if it is, say 10 thousand or more, the task becomes difficult, because defocus and motion blur make many images look alike.

3.2. Generative learning

In order to solve the task, we introduce generative learning, with a special attention to defocus and motion blur. Figure 2 shows the proposed way of generative learning. An

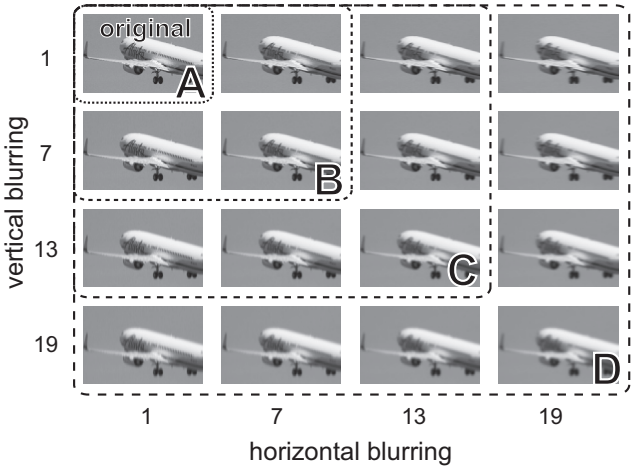
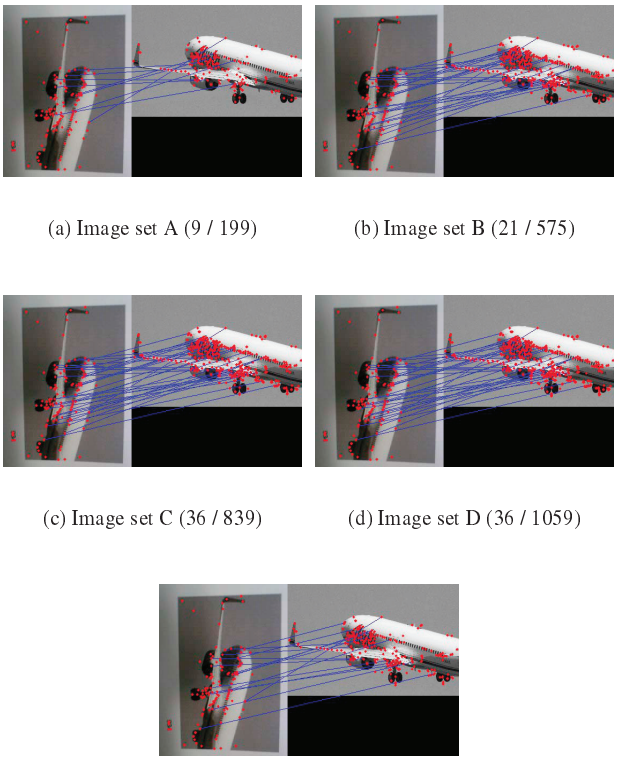


Figure 2. Generated blurred images.

original image A is blurred by changing the standard deviations of Gaussian. In Fig. 2, the parameter w of vertical and horizontal blurring such as 7, 13 corresponds to the standard deviation σ as $\sigma = 0.3(w/2 - 1) + 0.8$. By changing the parameter of horizontal and vertical blurring independently, motion blur can be simulated.

Figure 3 illustrates effects of generative learning. In each small figure (a)–(e), the left images correspond to a query image and the right images indicate its equivalent in the database. As the number of blurred images increases, more local features are obtained. This gains the number of matches, where matching is done by nearest neighbor search. Note also that even with the diagonal set $\mathrm{D}_{\mathrm{diag}}$ of Fig. 2, which simulates only defocus, the number of correct matches increases.

As shown in this figure, the advantage of generative learning is obvious: it allows us to increase the number of correct matches, which result in improving the recognition rate. Its disadvantage is also clear: additional local features cause problems on both memory and processing time.



(a) Image set A (9 / 199)

(b) Image set B (21 / 575)

(c) Image set C (36 / 839)

(d) Image set D (36 / 1059)

(e) Image set $\mathrm{D}_{\mathrm{diag}}$ (16 / 342)

Figure 3. A query image captured by a camera phone which includes 134 local features is shown on the left of each small figure. Image sets A–D correspond to blurring A–D shown in Fig. 2. $\mathrm{D}_{\mathrm{diag}}$ indicates the blurred images consisting of diagonal elements in Fig. 2. Local features extracted from blurred images are shown on the right of each small figure. Two numbers in the parentheses indicate the number of matches and the number of local features from blurred images, respectively.

4. Cascaded recognizers

4.1. Strategy for solving the problems

In order to employ generative learning, we need to solve the problems on memory and processing time without losing the recognition rate. In general, these factors are three-cornered: improvement on one factor may deteriorate others. In other words, it is required to have a methodology to balance the factors depending on a task.

As stated above, generative learning gives us a mean to improve the robustness, i.e., to improve the recognition rate for low-quality images. The parameter here is the number of generated images.

For solving the problem on memory, we employ a method of scalar quantization proposed in [4]. We should be careful that the scalar quantization lowers the discrimination power of original feature vectors, which may result in spoiling nearest neighbor search. We experimentally evaluate whether or not scalar quantization spoils the accuracy gained by generative learning.

Another problem is how to achieve the improvement on processing time. A powerful tool for the improvement is approximate nearest neighbor search. A parameter of approximation allows us to control the quality of resultant nearest neighbors, where the quality is defined based either on how far the resultant vector is as compared to the correct nearest neighbor, or how probable the correct nearest neighbor is obtained. In either cases, low-quality search with stronger approximation enables us a drastic speed-up (several orders of magnitude). The question is how to determine the value

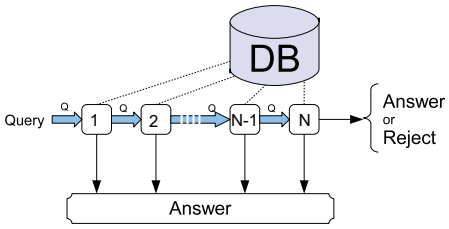


Figure 4. Cascaded recognizers.

of approximation parameters.

The problematic part of parameter definition is that an appropriate parameter value depends heavily on images to be recognized. Generally speaking, texture rich images are easy to recognize, while vague images are hard. If we would like to keep the recognition rate at a certain level, we need to recognize some "hard" images by setting the parameter for them. This indicates that unnecessarily long processing time is spent for many "easy" images.

In order to solve this problem, we propose a method to control adaptively an approximation parameter depending on images. The key technology is a *cascade* of recognizers. The advantage of our method is the guarantee of nothing getting lost by the cascade; The worst case computational cost of the cascade is equivalent to the cost without it.

4.2. Architecture and requirements of the cascade

The architecture of the cascade is shown in Fig. 4. Squares numbered by steps 1,...,N represent basic recognizers with approximate nearest neighbor search.

A basic recognizer takes as input a set of local feature vectors, casts votes based on matching them to obtain the result of recognition. To be precise, for each feature vector extracted from a query image (query local feature), the basic recognizer finds its corresponding local feature vector from the database (DB local feature) by applying approximate neighbor search. Since each DB local feature has a label of image, a matched query feature obtains the label for casting a vote. The recognition result is defined as the image with the maximum votes.

In the cascade, all basic recognizers are different as follows. The earlier the step s ($1 \leq s \leq N$) is, the stronger the approximation is applied. The recognition process is as follows. Firstly, the basic recognizer of the first step, i.e., the recognizer with the largest level of approximation, attempts to recognize a query image. If enough evidence is obtained at this step, the recognition process is immediately terminated and the result is outputted. Otherwise, a set of query feature vectors are sent to the next step to process the query image by a basic recognizer with less approximation. If enough evidence cannot be obtained by the last step N, the cascade outputs either the image with the maximum votes, or reject the query image.

Since "easy" images are recognized at early steps, the average speed can be improved. For the "hard" images the cascade can take its time to make a decision. Thus it is far different from the well-known cascade by Viola and Jones [10], which is for rejection; trying to reject meaningless regions as early as possible. On the other hand, the cascade proposed here is for recognition; trying to find easily recognizable images as early as possible.

Requirements for realizing the proposed cascade are: (1) how to make a decision of terminating the recognition process, (2) how to keep the efficiency even for "hard" images. For the first requirement, the computational cost for the decision should be negligibly small. For the second requirement, it is necessary to guarantee the computational cost of the worst case. Ideally, the computational cost of applying a series of recognizers 1,...,N is equal to the cost of applying the single recognizer which has the same approximation level at the step N.

4.3. Termination of recognition

It can generally be said that "hard" images have the following properties: (1) they often have less votes even if they are correctly recognized, (2) they have votes whose number is close to the image ranked at the second maximum votes. These tendencies can be used to define termination conditions. Let $v_1(s)$ and $v_2(s)$ be the numbers of votes for the first and second best images at the step s. Application of the cascaded recognizers is terminated at the step s if the following two conditions are satisfied:

$$v_1(s) > t, \tag{1}$$

$$r v_1(s) > v_2, \tag{2}$$

where t and r are parameters.

4.4. Efficient recognition for "hard" images

The performance of a basic recognizer is determined by its approximate nearest neighbor searcher. Consider N approximate nearest neighbor searchers (ANNSs) $1, ..., N$ whose level of approximation is different in such a way that for all s an ANNS $(s-1)$ $(1 < s \leq N)$ is with a stronger level of approximation than an ANNS s. Let $P_i^{(s)}$ be a set of feature vectors obtained by an ANNS s for calculating distance to a query feature vector $\boldsymbol{q}_i$. In general, an ANNS with a stronger level of approximation obtains a smaller number of feature vectors for distance calculation, i.e., $\forall i \; \forall s \; |P_i^{(s)}| \geq |P_i^{(s-1)}|$.

We define the following two property of an ANNS.

Definition 1 (monotonicity) *An ANNS is monotonic if the following equation holds:*

$$\forall i \; \forall s \; P_i^{(s)} \supseteq P_i^{(s-1)}. \tag{3}$$

Definition 2 (difference accessibility) *An ANNS has difference accessibility if it can efficiently access to elements in*

$$P_i^{(s)} - P_i^{(s-1)}. \tag{4}$$

If ANNSs in the cascade are monotonic, distance calculation at the step s is not necessarily for $P_i^{(s)}$ but for the difference $P_i^{(s)} - P_i^{(s-1)}$. This strategy enables us to equalize feature vectors for distance calculation with the cascade (in the steps from 1 to s) to those without the cascade (with only the ANNS s):

$$P_i^{(s)} = \bigcup_{k=1}^{s} \left(P_i^{(k)} - P_i^{(k-1)} \right) \tag{5}$$

where $P_i^{(0)} = \phi$. In addition, if ANNS holds the property of difference accessibility, the computational cost for accessing the set difference $P_i^{(s)} - P_i^{(s-1)}$ is negligible. Thus even for the worst case, i.e., the case in which the last ANNS N is applied, the computational cost is equivalent to a recognizer without the cascade.

The processing of recognition is as follows. Suppose we have already applied basic recognizers up to the step $(s-1)$. So far, we have already had, for each query feature vector $\boldsymbol{q}_i$, the *temporary* nearest neighbor $\hat{\boldsymbol{p}}^* \in P_i^{(s-1)}$. Thus at the step s, all we have to do is to calculate the distance to elements in $P_i^{(s)} - P_i^{(s-1)}$, and compare it with the distance to $\hat{\boldsymbol{p}}^*$ for updating the temporary nearest neighbor.

5. Details of Recognition

The proposed method has been built on the above notions of generative learning and the cascade. Basic recognizers in the cascade are the same as the one proposed in [4]. The details including those of basic recognizers are as follows.

5.1. Storage

PCA-SIFT is employed as local features. Original real-valued vectors are transformed into their scalar quantized form with 2bit/dim. Let $\boldsymbol{p} = (p_1, ..., p_n)$ be the scalar quantized version of PCA-SIFT feature vector where $p_j \in \{0, 1, 2, 3\}$. The probability of having each quantized value is kept equal by defining thresholds for quantization using a learning set. Since the average θ_j of the original values of each dimension j is almost 0, the quantized value 0 and 1 are for negative values and 2 and 3 for positive values.

Hashing quantized vectors is based on their bit-vector representation $\boldsymbol{u} = (u_1, ..., u_d)$ where

$$u_j = \begin{cases} 1 & \text{if } p_j - \theta_j \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

and $d < n$. The hash value of a feature vector $\boldsymbol{p}$ is then calculated as

$$H_{\text{index}} = \left(\sum_{j=1}^{d} u_j 2^{(j-1)} \right) \mod H_{\text{size}} \tag{7}$$

where H_{size} is the size of the hash table. Each feature vector is recorded in the hash table with its label (ID) of the image from which it is extracted. Collisions in the hash table are kept by the chaining method. If the length of the chain exceeds a predetermined threshold c, it is discarded and feature vectors can no longer be added to this bin. This processing is quite effective to cut the amount of memory and processing time.

5.2. Retrieval and Recognition

Retrieval of the nearest neighbor feature vector is also quite simple. For each query feature vector, the same processing of scalar quantization and calculation of a hash value is applied to look up the hash table. As a result a set X of candidate feature vectors with the same hash value are obtained.

At each basic recognizer in the cascade, the feature vector with the minimum distance

$$\boldsymbol{x}_* = \min_{\boldsymbol{x} \in X} \|\boldsymbol{x} - \boldsymbol{q}_i\| \tag{8}$$

is found for each query feature vector $\boldsymbol{q}_i$ for voting.

An obvious problem of such a simple method is that due to variation of imaging conditions, the query feature vector corresponds to a hash bin which is different from the bin of its nearest neighbor. A method for compensating it would be to perturb the bit vector. Let $\boldsymbol{q}_i = (q_1, ..., q_d)$ be a query feature vector and θ_j be the threshold for binarization of the j-th dimension. For the dimension j such that

$$|q_j - \theta_j| \leq e, \tag{9}$$

the nearest neighbor of $\boldsymbol{q}_i$ would have hashed into a different bin because q_j is close to θ_j, where e is a threshold. In such a case, not only the original bit vector but also the flipped bit vector with $u'_j = 1 - u_j$ is employed for accessing to the hash table to determine X as the union of retrieved feature vectors.

The maximum number of dimensions for the bit flip is predefined to b. The number of bit vectors produced by flipping b bits can be as large as 2^b. For example, if $b = d$, i.e., the number of dimensions of the bit vector, all feature vectors are in the set X. Thus the value of b should be limited to much less than d. If q_j of a query feature vector which satisfies Eq.(9) exceeds the limit b, dimensions with smaller indexes j are employed for flipping.

The parameter b is to control the search space and thus can be used to define the cascade as follows. At the first step

Table 1. Training sets.

Training set	# of images (ratio to A)	Ave. # of feature vectors (per image) [ratio to A]
A	1	5.0×10^2 [1.0]
B	4	1.6×10^3 [3.2]
C	9	2.6×10^3 [5.2]
D	16	3.3×10^3 [6.6]
D_{diag}	4	9.8×10^2 [2.0]

of the cascade, no bit flip is applied. At the next step, one bit flip is applied and at the step N, b bit flips are applied (thus, $N = b + 1$). Note that two properties, the monotonicity and the difference accessibility are both satisfied.

6. Experiments

6.1. Experimental settings

The proposed method was evaluated with the following experimental settings.

As images in the database we employed 1 million images obtained from Flickr using keywords such as "animal", "birthday", "food", as well as an upload date such as "2007.01.01". These images were shrunk to have their longest side less than or equal to 320 pixels, and stored in the database. Examples are shown in Fig. 5. Generative learning was applied to produce blurred images such as shown in Fig. 2. Table 1 shows the number of training images employed for extracting feature vectors including originals (the set A), as well as the average number of extracted feature vectors. Since less number of feature vectors are extracted from more blurred images, the increase of the number is less proportional to the number of added images.

Query images were prepared for recognition as well as rejection. Randomly selected 1,000 images were printed out in color on A4 paper with the size of either 16 images/page (1/16), or 4 images/page (1/4). Each image in these printouts was captured by 10 persons with different camera phones with or without the macro mode. Note that without the macro mode, there is little chance to obtain focused images for small printouts (1/16). As query images, in total 8,000 images (by 8 persons) for recognition and 2,000 images (by 2 persons) for rejection were prepared. The size of query images was fixed to QVGA (320×240).

The size of the hash table was set to $H_{\text{size}} = 2^d$. Unless otherwise noted, the following parameters were fixed to $b = 10$, $c = 100$, $d = 28$, $e = 400$, $t = 4$ and $r = 0.4$. In the following, processing time means time needed for recognizing one query image, excluding the time for local feature extraction. The computer employed for experiments was with AMD Opteron CPUs 2.8GHz (single CPU was employed for experiments) and 64GB memory.

Figure 5. Example images in the database.

6.2. Effects of generative learning

First of all, we evaluated the effect of generative learning for recognition without scalar quantization (16 bit/dim.), the cascade or rejection. The recognizer without the cascade is the application of a single basic recognizer with the parameter b. The representation without the scalar quantization is to describe original feature vector with 16 bit/dim [1]. The size of the database employed as original (A) in Fig. 2 was 10,000 images.

Table 2 shows the results. As the number of training images grew, the recognition rate improved from 81% (with the training set A) to 93.3% (with the training set D, +12.3%). The more the number of feature vectors is, the higher the recognition rate is. The biggest improvement was achieved for query images taken without the macro mode. For such low-quality queries, generative learning is quite effective to improve the recognition rate. With the training set D_{diag}, we still obtained progress, though the recognition rate was less than that with the training set D.

As for the processing time, recognition with the training set C doubled it from that with the training set A. The total memory needed for the recognition was 2.5GB (A), 3.5GB (B), 4.3GB (C), and 4.5GB (D). Thus the improvement on recognition rate was built on the sacrifice of both processing time and memory.

Figure 1 illustrates query images which failed to recognize with the training set A but succeeded with the training set C. From this figure, it is observed that the proposed method is effective to such severely blurred images.

6.3. Effects of the cascade

Next we introduced the cascade with the same parameter values of recognizers and the database. The rejection and scalar quantization were not introduced to this experiment. The results are also shown in Table 2. As compared to the results without the cascade, processing time was significantly improved with little loss of recognition rates. The most remarkable point is that the processing time remains

[1] Although the original feature vector is represented as 32 bit/dim., values are almost within the range of 16bit/dim. Thus for the fairness, original feature vectors are represented by 16bit/dim.

Table 2. Recognition rate[%] and processing time [ms] for each training set for the DB of 10,000 images and query sets.

cas-	DB	A(original)		B		C		D		D_{diag}	
cade	Query	recog. rate	time	recog. rate	time	recog. rate	time	recog. rate	time	recog. rate	time
w/o	ave.	81.0	7.7	89.9	12.0	92.6	14.8	93.3	16.4	91.0	9.5
w	ave.	81.0	2.3	89.9	1.7	92.5	1.5	93.2	1.5	90.9	1.6

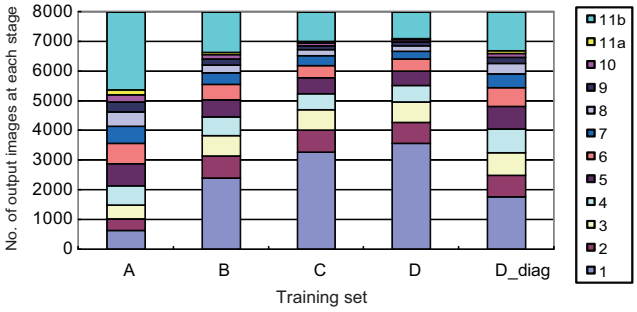


Figure 6. The number of images outputted at each stage.

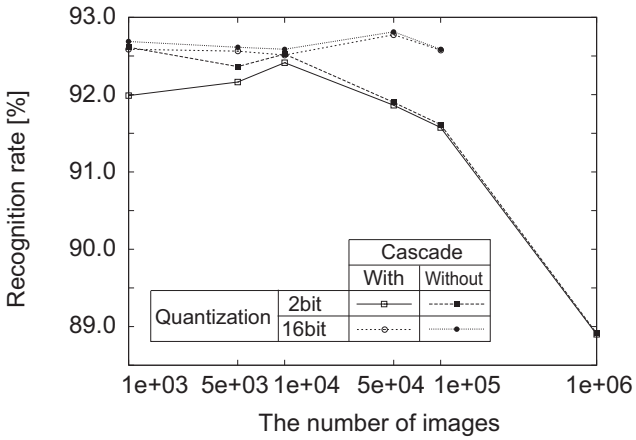


Figure 7. Recognition rate and the size of DB.

unchanged or even slightly decreases despite a significant increase of the number of local feature vectors in the training sets. As already shown in Table 1, the training set D includes vectors six times more than the original set A.

Figure 6 unveils what happened in the cascade. Each bar in the figure corresponds to a training set, and represents at which stage of the cascade the recognition terminated. As shown in Fig. 4, The number 1 indicates the first step. The numbers 11a and 11b represent the output of the last step: at the last step, the output which satisfies the termination conditions (the down-pointing arrow in Fig. 4) corresponds to 11a, and the rest (the right-pointing arrow) corresponds to 11b. From this graph, it is clear that the increase of feature vectors allows us to terminate the recognition at earlier steps. Since "easy" images are recognized at earlier steps, additional feature vectors obtained by the generative learning make the recognition of some images easier.

Although it may sound like a paradox, it has been shown that more feature vectors enable us to shorten the processing time. This phenomenon can be explained as follows. Additional feature vectors raise the chance that a query feature vector has its nearest neighbor in the same or close bins of the hash table. This helps us to accumulate correct votes at early steps of the cascade, and thus results in terminating earlier.

6.4. Rejection

The next experiment is on the effectiveness of rejection. As described in Sect. 1, it is better for users to receive the result of "rejection" than to have an erroneous result. Thus we tested whether the proposed method can be used as near error-free recognition at a certain level of rejection. The DB of size 10,000 images and the training set C were employed with scalar quantization of 2 bit/dim. Evaluation criteria are as follows. Let C_1, E_1, and R_1 ($C_1 + E_1 + R_1 = 1$) be the recognition rate, the error rate, and the rejection rate on queries for recognition, respectively, and E_2 and R_2 ($E_2 + R_2 = 1$) be the error rate and the rejection rate on queries for rejection, respectively. Parameter values were tested for the following ranges: $b = 5, 10, 15$, $c = 2, 5, 10, 100$, $d = 20, 24, 28$, $e = 200, 400, 600$, $r = 0.2, 0.4, 0.6$, $t = 4, 8, 12$. The best combination of parameter values which maximized each evaluation criterion shown in Table 3 was selected and applied based on 10-fold cross validation. Results are also shown in Table 3. Longer processing time was needed for queries for rejection, since they were rejected at the last step of cascade. For example, as shown in (2) of Table 3, the proposed method was successful to achieve the error rates E_1 and E_2 less than 1% by allowing the rejection rate $R_1 = 10\%$ and 10 ms processing time.

6.5. Scalability

Finally, the scalability of the proposed method was tested using the DB up to 1 million images. The proposed method was employed with and without the cascade, as well as with and without scalar quantization of 2 bit/dim. The training set used in this experiment is C.

For the recognition of 100K images, the total amount of memory consumed by the method with and without the

Table 3. Results with rejection.

criteria (for learning sets)		Queries for recognition				Queries for rejection		
		$C_1[\%]$	$E_1[\%]$	$R_1[\%]$	time [ms]	$E_2[\%]$	$R_2[\%]$	time [ms]
(1)	$\min(E_1+E_2+R_1)$, time $\leq$ 1ms	84.6	1.0	14.4	0.5	3.9	96.1	0.8
(2)	$\min(E_1+E_2+R_1)$, time $\leq$ 10ms	88.1	0.4	11.5	1.3	0.9	99.1	5.5
(3)	$\min(E_1+E_2+R_1)$, time $\leq$ 100ms	90.1	0.3	9.6	13.8	1.7	98.3	68.8

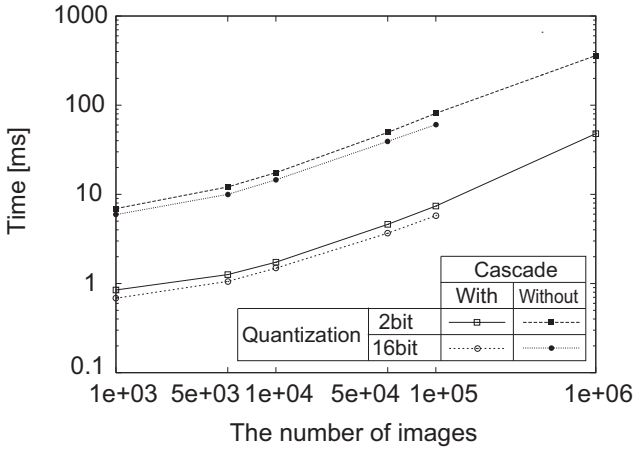


Figure 8. Processing time and the size of DB.

scalar quantization was 6.7GB and 22.6GB, respectively. Although the database of size 1 million images cannot be handled by the method without scalar quantization due to the limit of available memory (64GB), the method with scalar quantization worked with 31.6GB memory.

Figure 7 shows the relationship of the size of DB to the recognition rate. As shown in this figure, scalar quantization of 2 bit/dim. deteriorated the recognition rate about 1%. On the other hand, it is turned out that the cascade has no harmful effect.

Figure 8 shows the relationship between the recognition rate and the processing time. It is clearly shown that the cascade enables us to speed up the processing 10 times. The scalar quantization slightly increased the processing time for the processing of quantization.

From the viewpoint of scalability, both the cascade and the scalar quantization deserve to be used because: (1) the cascade allows us significant speed up (10 times), (2) the scalar quantization enables us to reduce the amount of memory to less than 30% of the original with a little loss of the recognition rate (1%).

7. Conclusion

We have proposed a method of specific object recognition for low-quality images. Generative learning by various Gaussian blurring is helpful for improving the recognition rate. The burden caused by generative learning is overcome by two methods, scalar quantization and the cascade of recognizers based on approximate nearest neighbor search. From the results of various experiments, the proposed method has proven to be effective, robust and efficient. The most important fact found through the experiments is that in order to cut the processing time it is effective to increase the number of feature vectors for indexing images.

Future work includes further expansion of scale as well as application of the proposed method to 3D object recognition.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Comm. of the ACM*, 51(1):117–122, 2008.

[2] H. Ishida, T. Takahashi, I. Ide, Y. Mekada, and H. Murase. Identification of degraded traffic sign symbols by a generative learning method. In *Proc. ICPR2006*, pages 531–534, 2006.

[3] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR2004*, volume 2, pages 506–513, 2004.

[4] K. Kise, K. Noguchi, and M. Iwamura. Memory efficient recognition of specific objects with local features. In *Proc. of the 19th International Conference of Pattern Recognition (ICPR2008)*, 2008.

[5] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[6] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. CVPR2006*, pages 775–781, 2006.

[7] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE PAMI*. Accepted for publication.

[8] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE PAMI*, 19(5):530–535, 1997.

[9] J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proc. of ICCV2003*, pages 1470–1477, 2003.

[10] P. Viola and M. Jones. Robust real-time object detection. In *Second Int'l Workshop on Statistical and Computational Theroies of Vision – Modelling, Learning, Computing, and Sampling*, 2001.