11

Image Kernel for Recognition

Zhu XiaoKai and Li Xiang
National University of Defence Technology
P.R.China

1. Introduction

Kernel-based methods, such as SVM classifier, have been proven to have more predominance of generalization and better performance of classification than traditional methods. As the main point of these technologies, kernel functions can increase the computational power of traditional linear learning machines by projecting the data into a high dimensional feature space, and can transform a non-linear problem into a linear problem (John, S.T. & Nello, C. 2004).

Although kernel functions have been widely used in pattern recognition, they have some weaknesses. Traditional kernel functions only accept one-dimensional vector as their input data. But some real-world data such as image data are often two-dimensional matrices, which can not be directly accepted by the kernel functions unless doing some preprocessing work. One way is to abstract features. One or more features that can denote some information of the image object are combined into a one-dimensional vector, and then a two-dimensional problem becomes a simple one-dimensional problem. This is the common way to do with the image objects. There are many categories of ways to abstract features (Sergios, T. & Konstantinos, K., 2006), including invariant moment, PCA, ICA, statistic analysis, texture analysis, shape analysis, etc, which are not introduced in detail in this paper. But every feature can only be efficient to some special object. How to select the appropriate ones is always a difficult problem. The other way is to decrease the dimensions of data. The simplest method is to treat the image data as a one-dimensional vector. C. Kaynak (1995) divides 32× 32 bitmaps of handwritten digits into nonoverlapping blocks of 4 × 4 and counts the number of on-pixels in each block. Then he gets a vector of 64 elements and uses it as the feature vector. This way can only be efficient with data of small size. In fact, many statistic analysis approach of the first way do the same thing. They treat the image as a set of non-relevant pixels and the structural information of two-dimensional data are lost.

In this chapter, we propose a new kind of kernel function that can directly accept image data as input data. Section 2 introduces the traditional RBF kernel function in brief and educes our idea. In Section 3, we describe our kernel function in detail, which is based on the RBF kernel function. In Section 4, the new kernel function is compared with the old approaches on UCI Optical Handwritten Digits dataset and COIL dataset.

2. Traditional Kernel Functions

Our idea comes from the traditional kernel Functions. First, let us see the three types of old ones which are generally used.
Polynomial:

$$K_{poly}(u,v) = (\sigma u \cdot v + r)^d \tag{1}$$

RBF:

$$K_{rbf}(u,v) = \exp(-\frac{\|u-v\|^2}{\sigma^2}) \tag{2}$$

Sigmoid:

$$K_{sig}(u,v) = \tanh(\sigma u \cdot v + r) \tag{3}$$

Here, u and v are one-dimensional vectors, σ and r are parameters. Out purpose is to construct a new form of kernel function where u and v can be two-dimensional data.
In (1) and (3), the operation between u and v is inner product, which can not operate on image data generally. And we don't pay attention to them.
In (2), the operation is $\|u-v\|$, which always indicates the distance between two vectors. And we know that distance between two objects can be considered as the similarity of them. So we get an idea that if the distance or similarity of two image data can be calculated, we can use it to replace the $\|u-v\|$ and the new RBF kernel function can be written as (4).
RBF2D:

$$K_{rbf\,2D}(A,B) = \exp(-\frac{d^2(A,B)}{\sigma^2}) \tag{4}$$

Here, A and B are two-dimensional data of target in images, and $d(A,B)$ is the distance or similarity of them. RBF2D is the new kernel function proposed in this paper that can accept image data.
The particular expression of RBF2D and $d(A,B)$ will be introduced in Section 3.

3. RBF2D

Note that in (4) A and B only appear in the form of $d(A,B)$, so before we get the expression of RBF2D, we should get $d(A,B)$ at first.

3.1 Distance between Image Data

Image data are in the form of matrix. The distance between two matrices can be computed using Frobenius Norm (Horn R.A. & Johnson C.R., 1985) generally.

$$\|A-B\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(a_{i,j}-b_{i,j})^2} \tag{5}$$

Here, A and B are $M \times N$ matrices. And From the formula, we can see that, all elements in A and B are out-of-order, only statistical information is reserved. There is one serious problem with this method. Unlike in a vector or a data set, an element in a matrix has relation to not only the one on the left side and the one on the right side, but also those above it and below it, even those in other directions, as be shown in Figure 1.

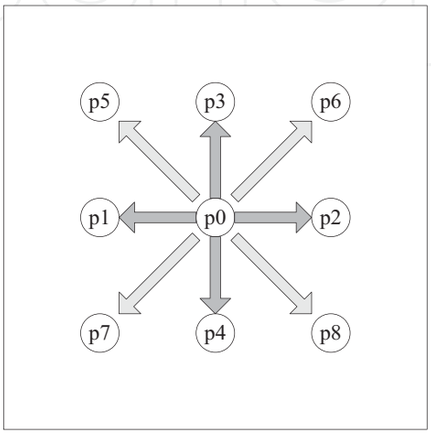


Fig. 1. An element p0 in a matrix and its neighbors p1~p8

Especially for image data, most targets are objects of some shape or structure and cover a region in the image. Structural information is the same important as the statistical one. So we should define a new form of $d(A,B)$ which involves structural information.

Zhou Wang (2004) proposes SSIM. He uses correlation between the two images to quantify the structural similarity. Luminance, contrast, and structure information are included in SSIM. The result shows that SSIM is efficient in quantifying the visibility of differences between a distorted image and a reference image. But the image is treated as a whole entity in SSIM. This will make the algorithm unstable in the following conditions.

Fig. 2. The same object under different luminance conditions

The two images in Figure 2 are the same object under different luminance conditions. Because the object has an anomaly surface, they look different in luminance, and SSIM doesn't work. For solving this problem, we propose our method which is block-based. Although the two images look different in every block at the corresponding position (in Figure 3), we can increase their similarity after some simple preprocessing work, which will not work on the whole image.

Fig. 3. The same object under different luminance conditions with blocks

So, out definition of $d(A,B)$ based on blocks has the following form, in (6).

$$d^2(A,B) = \sum_{n=1}^{N_{block}} \|A_n - B_n\|_F^2 \cdot \omega_n \tag{6}$$

Here, A_n and B_n are the data matrices of the nth block. ω_n is the weight of $\|A_n - B_n\|_F^2$ which can involve the luminance, structure information. This is the main part of our work, and we will discuss it in the following sections. N_{block} is the count of the blocks.

The expression is similar to $\|u - v\|$, because it will be used in RBF2D which is based on RBF kernel function (in Equation 7). We make $\|A_n - B_n\|_F^2$ the main part of it and other information as the weight.

$$K_{RBF2D}(A,B) = \exp\left(-\frac{\sum_{n=1}^{N_{block}} \|A_n - B_n\|_F^2 \cdot \omega_n}{\sigma^2}\right) \tag{7}$$

Weight ω_n is the combination of luminance difference weight, content difference weight, self complexity weight, and position weight. The Following sections will discuss each part of $d(A,B)$ in detail.

3.1.1 $\|A_n - B_n\|_F^2$

This part has the same computational formula (in Equation 5) as $\|u - v\|$. For image targets, it indicates the energy difference and it is the base of our distance measure.

3.1.2 Luminance Difference Weight.

We suppose C_n is the luminance difference of A_n and B_n.

$$c_{i,j} = a_{i,j} - b_{i,j} \tag{8}$$

Then we suppose that if the two images are similar, their luminance difference will be a smaller value than that of two images which are not alike. The mean value of C_n

$$\mu_c = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} c_{i,j} \tag{9}$$

can estimate the degree of how much their luminance difference is. If they are equal, $\mu_c = 0$; else, $\mu_c \neq 0$. The bigger the absolute value of μ_c is, the much difference the two images have. Then the weight ω_{lum} is a function of μ_c.

As a weight function, we hope that its value is between 0 and 1, and when the two images have the same luminance level, its value is 0. So we give the following expression

$$\omega_{lum} = 1 - \exp(-\frac{|\mu_c|}{C_1}) \tag{10}$$

where the constant C_1 is included to avoid ω_{lum} increasing too fast. For 8-bit grayscale or 24-bit true-color images, the maximum of each pixel is 255, we choose $C_1 = 10\sim20$ which is calculated using the image data sets on internet.

3.1.3 Content Difference Weight.

For images, content is more important than luminance. The two images in Figure 2 have different luminance level, but obviously they are the same object. We consider that the luminance difference between two images of the same object will be less complex than that of different objects. We suppose C_n is the luminance difference of A_n and B_n. And the standard deviation of C_n

$$\sigma_c = \left(\frac{1}{M \times N - 1}\sum_{i=1}^{M}\sum_{j=1}^{N}(c_{i,j} - \mu_c)^2\right)^{\frac{1}{2}} \tag{11}$$

can estimate the degree of how much the difference is.
As in Section 3.1.2, we give the following expression

$$\omega_{con} = 1 - \exp\left(-\frac{\sigma_c}{C_2}\right) \tag{12}$$

where the constant C_2 can avoid ω_{con} increasing too fast. For 8-bit grayscale or 24-bit true-color images, we choose $C_2 = 50\sim70$ which is calculated using the image data sets on internet.

3.1.4 Self Complexity Weight.

We consider that if an image is complex itself, it will be harder to find a similar one to it. In simple object while A'_n and B'_n are complex ones. We think the distance between A'_n other words, suppose two groups of images $\|A_n - B_n\|_F^2 = \|A'_n - B'_n\|_F^2$, A_n and B_n are two and B'_n is smaller. So

$$\omega_{complex} = \exp\left(-\frac{\sigma_{A_n}}{C_3}\right) \cdot \exp\left(-\frac{\sigma_{B_n}}{C_3}\right) \tag{13}$$

Where σ_{A_n} and σ_{B_n} can estimate their complexity, and C_3 is same as C_1, C_2. We choose $C_3 = 40\sim60$ for 8-bit grayscale or 24-bit true-color images.

3.1.5 Position Weight.

First, We think that the position of each block has relation to its contribution for the distance $d(A,B)$. The block close to the center will have greater weight. Second, the target is generally at the center of the image, while there may be some background objects around it. The block close to the edge may have more background information. To avoid these blocks' effect, they will be set smaller weight. So we hope ω_{pos} has Gauss form (in Figure 4, Equation 14), where the constant s is the value that is predefined for the block at the corner. r is the distance between the block and the center. r_0 is half of the diagonal length. We choose s=0.5 in our study.

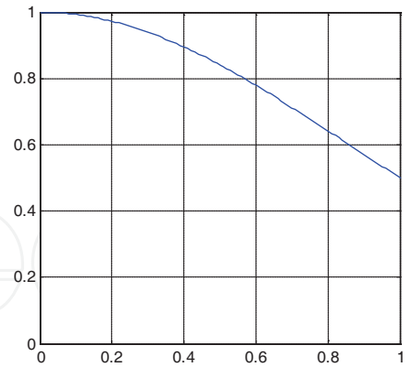


Fig. 4. The Relationship between Position Weight and its Position

$$\omega_{pos} = \exp(\log s \cdot (\frac{r}{r_0})^2) \tag{14}$$

Finally, all parts of ω_n are ready and we give

$$\omega_n = \left(m \cdot \omega_{lum} + (1-m) \cdot \omega_{con}\right) \cdot \omega_{complex} \cdot \omega_{pos} \tag{15}$$

where parameter m (0<m<1) is the weight of ω_{lum} and ω_{con}. Because we think ω_{lum} is from the whole view while ω_{con} is from the detail view. Parameter m is set to control the proportion how much they contribute to ω_n.

From all above discussions in Section 3.1, we know that

$$0 < \omega_n < 1 \tag{16}$$

It can be used as a weight. Then our $d(A,B)$ is accomplished finally.

3.2 Blocking Option

The size of each block and how the blocks are organized are also important to our RBF2D kernel function. But we will not discuss them in detail in this chapter. We only give our solution in this paper.

3.2.1 Size

We find that the size of each block can have effect on the performance of our image kernel. As shown in figure 5, if the blocks are too big, then the blocking operation is nonsense because the problem that we want to avoid when using the whole image will be met again. On the other side, if the blocks are too small, they can not contain the structural information that we hope they would have done.

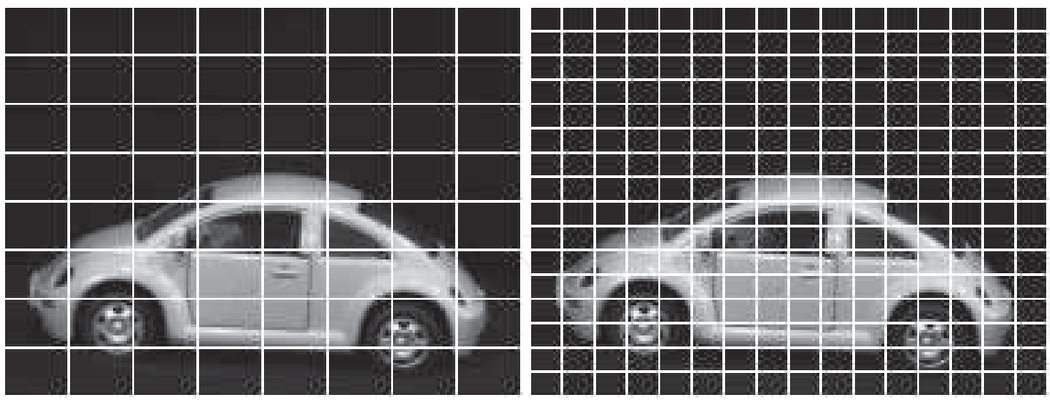


Fig. 5. Blocking mode which have different sizes

So, the size of block will be decided by the minimun shape in the image that can have some information.

3.2.2 Organization

We propose three image kernels based on their different organization forms of blocks.

(1) Normal Image Kernel. Blocks are organized as shown in figure 5. This is the simplest way. But its calculation efficiency is low and will be unstable when the edge of some block happens to overlap with that of the object.

(2) Redundant Image Kernel. We add redundancy part between two neighbors based on Normal Image Kernel. This image kernel is more stable but its calculation efficiency is even lower.

(3) Discrete Image Kernel. We use ROI (Regions of Interest) technology to decide some discrete blocks while other regions of the image will be ignored. This method has many advantages. First, the number of blocks which are calculated is small than the other two image kernel, its calculation efficiency is high. Second, the position of each block on one image is decided by its information distribution. The corresponding blocks on the other image is decided using image matching processing. So the positions of the object in the two images can have some difference, while the same thing will reduce the performance of the other two image kernels. The experiment result shows that the Discrete Image Kernel can work on the images in which the object can have different stances or be sheltered partly.

4. Experiment Results

4.1 UCI Optical Handwritten Digits Dataset

This dataset include about 5500 normalized bitmaps of handwritten digits gathered from 43 different people. Image size is 32×32 .

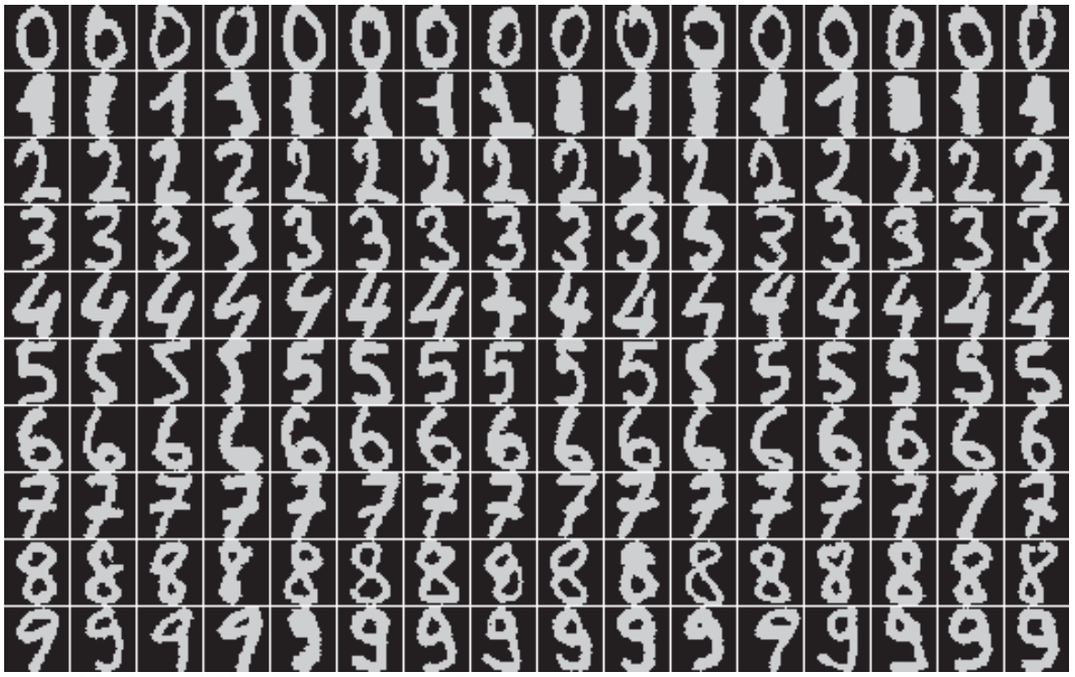


Fig. 6. UCI Optical Handwritten Digits Dataset

C. Kaynak divided these images into blocks of 4×4 and counted the pixels in each block. Then he got an input matrix of 8×8. His classifier was KNN based on Frobenius Norm and the final ratio of recognition is 97%~98% for each digits.

We choose digit '1' and digit '7' as our targets because they have similar appearance. 20% of total 400 samples are training set, others are testing set. The experiment will be repeated 10 times, and training set was selected randomly. And because the images are 1-bit bitmaps, we choose C_1=1, C_2=1, C_3=1, m=0.5, s=0.5. Blocks' size is 4×4. And finally our ratio of recognition is 99%~100%.

Fig. 7. SVs of RBF2D-based SVM

Fig. 8. Some samples which are difficult to recognized for Kaynak's method

From Figure 7 and Figure 8, we can see that the samples which are difficult to recognized for Kaynak's method are always the SVs of our SVM based on RBF2D. So our ratio can reach 100% when they are all in train set and treated as SVs.

4.2 COIL dataset

COIL data set includes 1000 objects and each object has 24 8-bit grayscale images under different luminance conditions. The size of each image is 144×192. We choose 3 objects. They are shown in Figure 9.

Fig. 9. Three Objects from COIL Data set

There are not enough samples. We only use our kernel function to calculate the gram matrix of the objects, because the learning mechanism of kernel-based classifiers is generally based on it. In our experiment, we set C_1 =20, C_2 =70, C_3 =60, m=0.5, s=0.5. Blocks' size is 24×24. The gram matrix is shown in Figure 10.

There are totally 72 samples, 24 for each object. From figure 6, it is obvious that only the two sample of the same object have the value close to 1, while others are close to 0. A kernel-based classifier (like SVM) can easily find the SVs of each class using linear programming algorithm or quadratic programming algorithm.

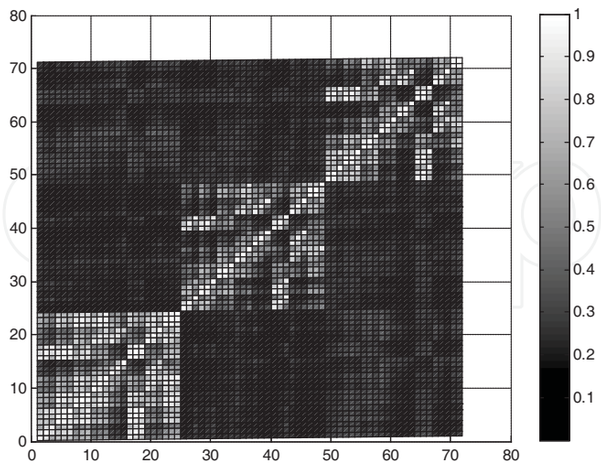


Fig. 10. The Gram Matrix of three Objects

4.2.1 COIL Objects with different angle of view

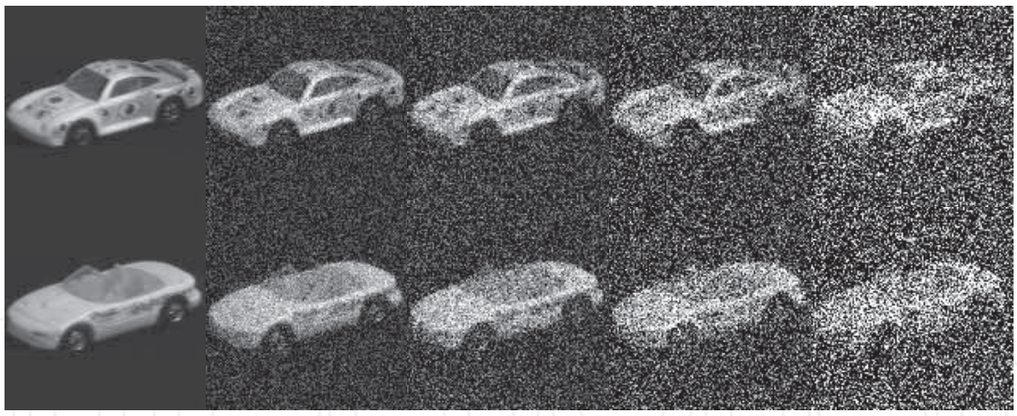
In Coil-100 dataset, the objects have different angles of view. We choose two as our targets.

(a) 315° (b) 320° (c) 325° (d) 330° (e) 335° (f) 340°

Fig. 11. The Two Objects With Different Angel of View

There are only 12 images of the two objects. To expand the sample size, we add +2,+1 displacement to each object at four directions (up, down, left, right). And we get totally 12× 25=300 samples. We select 50% as train set and the others as test set and use Redundant Image Kernel and Discrete Image Kernel introduced in section 3. Classifier is the standard SVM.

For testing the performance, we also add noise to the images.



(a) no noise (b) 10db (c) 0db (d) -10db (e) -20db

Fig. 12. Images with noise

The experiment results are shown in table 1.

	Redundant Image Kernel		Discrete Image Kernel	
no noise	100%	34%	100%	3.3%
10 db	100%	100%	100%	30%
0 db	90%	100%	96%	66%
-10 db	85%	100%	90%	100%
-20 db	47%	100%	58%	100%

Table 1. Result of experiment, left is ratio of recognition, right is SVs/Samples

First, let's see the ratio of recognition. Discrete image kernel is the same as redundant image kernel. They can work until the noise level is reduced below -20 db. As introduced in sections before, discrete image kernel only uses part of the image, so its speed is higher.
Second, let's see the numbers of SVs. For redundant image kernel, all samples become SVs when the noise level is 10db. In this condition, every sample is SV and have to be saved so as to be used in testing processing. The classifier will have bad performance. While the discrete image kernel can work until noise is reduced below -10db.
To explain this problem, Figure 13 shows gram metrices of two kernels when there is no noise.

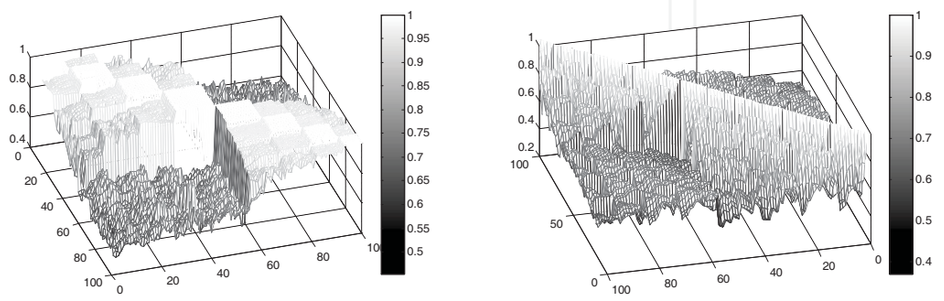


Fig. 13. Gram Metrix of two image kernel, left is Discrete Image Kernel, right is Redundant Image Kernel

From the gram metrix of two kernels, we can find that the difference of the two objects in discrete image kernel is more distinct and easier to use.

4.2.2 COIL Objects with sheltering

In this section, we will add some shelter to object images which are shown in figure 14.

Fig. 14. Images with sheltering, shelter ratio is 7.8%,15.6%,23.4%,31.3%,39.1%,46.9%(from left to right)

The methods and parameters are the same as section 4.2.1. We use Redundant Image Kernel and Discrete Image Kernel introduced in section 3. Classifier is the standard SVM.
Train set is the full object image as in 4.2.1, while we add sheltering on test set, and use the new test set to test the two image kernel.

For each shelter ratio, the experimint is repeated once.
The result is shown in table 2.

	7.8%	15.6%	23.4%	31.3%	39.1%	46.9%
Discrete Image Kernel	Yes	Yes	Yes	Yes	Yes	No
Redundant Image Kernel	Yes	Yes	No	No	No	No

Table 2. Result of experiment, yes = can work, no = can't work

We can find that the discrete image kernel can work unless the shelter ratio reaches 40%, while the redundant image kernel can only work under 20%.
For Redundant Image Kernel, the object is sheltered means that the image has changed. While for Discrete Image Kernel, the algorithm only use part of the blocks (in figure 15).

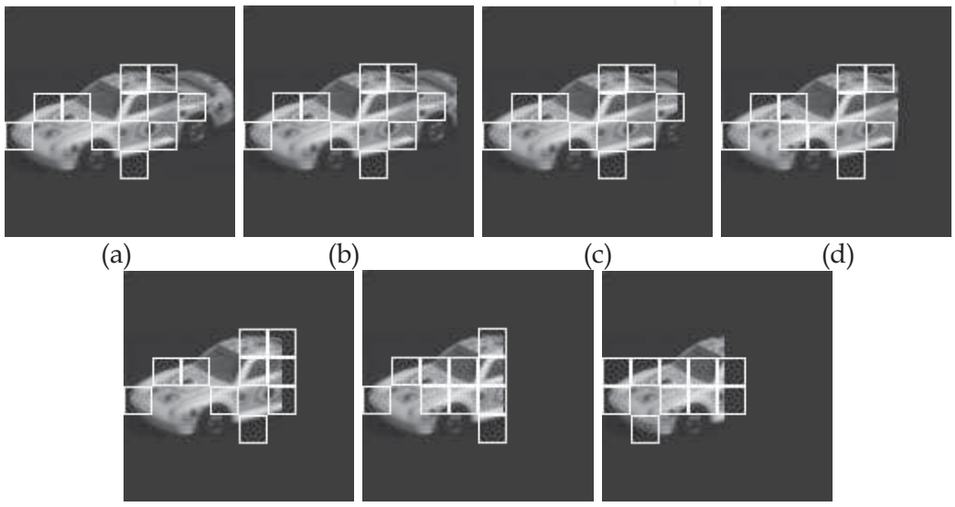


Fig. 15. The Selected Blocks in Discrete Image Kernel. shelter ratio is (a) 0%, (b) 7.8%, (c) 15.6%, (d) 23.4%, (e) 31.3%, (f) 39.1%, (g) 46.9%

The seven image in figure 15 can be grouped into four categories.

(1) Figure 15(a). There is no sheltering in this condition. So (a) can be treat as a reference.
(2) Figure 15(b)~(c). Although the object has been sheltered, but all the selected blocks are the same as (a), so Discrete Image Kernel can work without any performance decrease.
(3) Figure 15(d)~(f). Here the object is sheltered partly and the selected are the same as (a) mostly. Only 1~2 blocks are changed, and they can not affect the total result.
(4) Figure 15(g). Mostly a large number of selected blocks are changed, and the result is unstable.

5. Conclusion

In this chapter, we have summarized the deficiency of traditional kernel functions on image recognition and proposed the distance measure of images and RBF2D kernel function which can accept two-dimensional image data as input data without abstracting the features that we often do nowadays.

We compare them to the old method using the UCI Optical Handwritten Digits dataset. The result indicates that RBF2D have good performance on image target.
Also we do some experiment to test the new image kernel when object are viewed from different angle, with noise, and even sheltered. The results show that our new image kernel can work in all these conditions.

6. Reference

John, S.T. & Nello, C. (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press, 7-111-17853-X, Cambridge

Horn, R.A. & Johnson, C.R. (1985). *Matrix Analysis*. Cambridge University Press, Cambridge

Sergios, T. & Konstantinos, K. (2006). *Pattern Recognition Third Edition*. Elsevier Pte Ltd, 981-259-707-7, 978-981-259-707-6, Singapore

Zhou, W. ; Alan, C.B. ; Hamid, R.S. & Eero, P.S. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, Vol.13, No.4, pp.600-612

C. Kaynak. (1995). Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.

Pattern Recognition
Edited by Peng-Yeng Yin

ISBN 978-953-307-014-8
Hard cover, 568 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

For more than 40 years, pattern recognition approaches are continuingly improving and have been used in an increasing number of areas with great success. This book discloses recent advances and new ideas in approaches and applications for pattern recognition. The 30 chapters selected in this book cover the major topics in pattern recognition. These chapters propose state-of-the-art approaches and cutting-edge research results. I could not thank enough to the contributions of the authors. This book would not have been possible without their support.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen