Supplementary Materials: Kernel Sparse Representation with Pixel-level and Region-level Local Feature Kernels For Face Recognition

Cuicui Kang*, Shengcai Liao, Shiming Xiang, Chunhong Pan

1. Comparison with KMTJ

We compared the proposed KCDSRC algorithm with the KMTJSRC algorithm [3] on the Extended YaleB and the CMU-PIE databases. The proposed LBP$_h$-K_H kernel is used due to its best overall performance than the other kernels. The experiment is conducted under three conditions, illumination, random noise, and synthesized continuous occlusion, where the settings are the same as we have applied before. The KMTJ source code which is used in our experiment is available on the author's homepage[1]. The parameters of KMTJ used in the experiment are the same as the KCD. The recognition results along with their standard deviations are shown in Fig.s1. In each figure there are four curves, which stand for the KCDSRC method with LBP$_h$-K_H kernel on the Extended YaleB and the CMU-PIE (YaleKCD & PieKCD), and the KMTJSRC method with LBP$_h$-K_H method on the Extended YaleB and the CMU-PIE (YaleKMTJ & PieKMTJ) respectively.

On the left of Fig.s1, it is the recognition results of the KCDSRC and KMTJSRC algorithms tested on original images of the YaleB and the CMU-PIE databases. We can see that the performance on the CME-PIE is better than on the YaleB, which is also occurred in the illumination part subsection 4.2. And the reason lies in the stronger illumination changes on the YaleB database compared to the CMU-PIE (see Fig.6). It can also be observed that the KCDSRC with LBP$_h$-K_H kernel performs better than the KMTJSRC with LBP$_h$-K_H on both the YaleB and the CMU-PIE databases. With

*Corresponding Author. Email: cckang@nlpr.ia.ac.cn. Phone: +86 01062612582

[1] https://sites.google.com/site/xtyuan1980/publications.

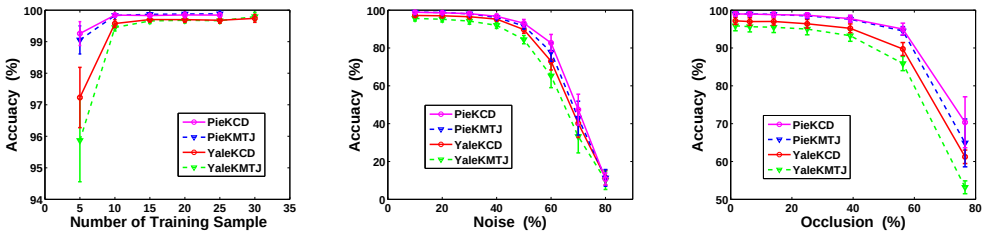

Figure s1: Comparison of KCD and KMTJ on the Extended YaleB and PIE databases with illumination (left), random noise (middle) and synthesized continuous occlusion (right).

the increasing number of the training samples, both the KCDSRC and the KMTJSRC achieve higher performances, and they are almost the same at last. In the middle of Fig.s1, experiments with noise show that he KCDSRC algorithm also outperforms the KMTJSRC algorithm. This is consistent with the experiments with occlusions, of which the performances are shown on the right of Fig.s1. Interestingly, the performance gain of KCDSRC compared to KMTJSRC is larger under more difficult settings, for example, smaller number of training samples, and larger occlusions.

2. LBP$_h$-K_H

p	q	p⊙q
1	1	1
0	0	1
1	0	0
0	1	0

p′	q′	p′*q′
1	1	1
-1	-1	1
1	-1	-1
-1	1	-1

Figure s2: XNOR truth table (left) and the corresponding results of the entry-wise multiplication (right) obtained from the modified encoding (replacing 0 by -1).

It should be pointed out that the proposed LBP$_h$-K_H is equivalent to the linear kernel when the linear kernel is applied to a modified LBP encoding where all 0's in the original LBP encoding are replaced by -1's. Fig.s2 shows the truth table of the XNOR operator used in K_H, and the corresponding entry-wise multiplication results obtained from the modified LBP encoding. Suppose n is the length of two binary strings $\mathbf{a}$ and $\mathbf{b}$ to be compared, and k bits of them are the same between the two binary strings. Then, we get the Hamming distance based kernel $K_H(\mathbf{a}, \mathbf{b}) = k/n$. For the linear kernel, from

Fig. s2 we know that the entry-wise multiplication of the two modified LBP encodings $\mathbf{a}'$ and $\mathbf{b}'$ results in k 1's and $(n-k)$ -1's. Hence, we get $K_{linear}(\mathbf{a}', \mathbf{b}') = 1/n \sum_i a'_i * b'_i = k/n - (n-k)/n = 2k/n - 1$. Therefore, we can infer that,

$$K_H(\mathbf{a}, \mathbf{b}) = \frac{K_{linear}(\mathbf{a}', \mathbf{b}') + 1}{2}, \tag{s1}$$

From Eq.(s1), we can see that the proposed K_H can be implemented by K_{linear}. However, K_H is more efficient than K_{linear} since the storage of binary bits is less than decimal numbers, and the most important, the XOR operator in K_H (see Eq.(16))is much faster than the multiplication operation in the K_{linear}. Therefore, computing the kernel matrix via the original K_H defined in Eq.(16) is better than via the K_{linear}.