Fast Kernel Sparse Representation

Hanxi Li[1,2,4], Yongsheng Gao [1,2], Jun Sun[3,4]
[1]NICTA, Queensland Research Laboratory, QLD, Australia
[2]Griffith University, QLD, Australia
[3]NICTA, Canberra Research Laboratory, ACT, Australia
[4]Australian National University, ACT, Australia

***Abstract*—Two efficient algorithms are proposed to seek the sparse representation on high-dimensional Hilbert space. By proving that all the calculations in Orthogonal Match Pursuit (OMP) are essentially inner-product combinations, we modify the OMP algorithm to apply the kernel-trick. The proposed *Kernel OMP* (KOMP) is much faster than the existing methods, and illustrates higher accuracy in some scenarios. Furthermore, inspired by the success of group-sparsity, we enforce a rigid group-sparsity constraint on KOMP which leads to a non-iterative variation. The constrained cousin of KOMP, dubbed as *Single-Step KOMP* (S-KOMP), merely takes one step to achieve the sparse coefficients. A remarkable improvement (up to $2,750$ times) in efficiency is reported for S-KOMP, with only a negligible loss of accuracy.**

I. Introduction

During the past decade, Sparse Representation (SR) (or sparse coding) has attracted much attention in computer vision community. Sparse representations approximate a input vector by using a sparse linear combination of *atoms* from an over-complete *dictionary*. In different scenarios, SR could solve various problems such as image annotation[1], image denoising[2], image restoration[3] and image classification[4], [5]. Some state-of-the-art performances were reported with SR approaches. To achieve higher classification accuracy, Gao *et al*. [6] equip sparse representations with the kernel trick[7]. In essence, they solve the SR problem in high-dimensional feature space, where the non-linearly mapped feature points with the corresponding labels are easier to separate. Gao *et al*. believe that the sparse representation will also benefit from the non-linear mapping. Their experiment proves this conjecture empirically.

The conventional way to solve the SR problem is interior-point based, *e.g*. basis pursuit[8]. However, it suffers from high computational complexity which makes it prohibitively expensive for massive dictionaries or real-time applications. The kernel sparse representation brings even heavier computational burden due to the extra calculation for kernel matrix. Instead of the standard approach, the authors of [6] employ a more recently-proposed method termed Feature-Sign Search (FSS)[9]. It is faster than interior-point approaches yet the implementation is complicated and no iteration-number bound is guaranteed. Both the above approaches achieve sparse representation via solving a ℓ_1-regularized Least Square (LS) which is a relaxed version of the original optimization problem. On the other hand, the Orthogonal Matching Pursuit (OMP)[10], [11] solves the original ℓ_0 minimization problem in a greedy fashion. It is extremely fast and could be used for real-time applications[12].

As proved in this paper, the OMP possesses an advantage that every calculation inside OMP could be expressed as a combination of inner products. With minor modifications, *We could directly perform the OMP procedure in the high-dimensional space*. In this paper, we derive the procedure of Kernel OMP (KOMP). The proposed algorithm outperforms other sparse representation solvers in terms of efficiency and signal-recovery accuracy. Additionally, for Sparse Representation Classification (SRC), an aggressive constraint is enforced to guarantee the rigid group-sparsity. This extreme constraint makes the KOMP non-iterative, *i.e. one could obtain the sparse coefficient in one step.* We gain a massive improvement in efficiency thanks to this constraint. The non-iterative KOMP, termed S-KOMP ("S" for Single-step), also achieves the best classification rates on low-dimensional face subspace.

The rest of this paper is organized as follows. We briefly review the related literature background in the next section. In Section III, we derive the OMP algorithm in the context of kernel method. We present the group-sparsity constraint and the S-KOMP algorithm in Section IV. Our methods is verified, by comparing with the state-of-the-art approaches in Section V. Conclusion and future work can be found in the last section.

II. Preliminaries

A. Sparse representation and its variations

The goal of sparse representation is to represent input vector $\mathbf{y} \in \mathbb{R}^d$ approximately as a weighted linear combination of a small number of "atoms" or "basis". The atom-set $\mathbf{X} \in \mathbb{R}^{d\times N}$ is dubbed "dictionary" which could be the collection of original data[4] or learned basis[13]. Given the prior knowledge that the combination coefficient $\boldsymbol{\beta} \in \mathbb{R}^N$ is sparse, one could achieve the sparse coefficient via

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_0, \quad \text{s.t. } \mathbf{X}\boldsymbol{\beta} = \mathbf{y}, \tag{1}$$

where $\|\cdot\|_0$ denotes the ℓ_0 norm. Since (1) is NP-hard [11], it is commonly relaxed to

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_1, \quad \text{s.t. } \mathbf{X}\boldsymbol{\beta} = \mathbf{y}. \tag{2}$$

which is Linear Programming (LP) and could be solved in the polynomial time. The relaxed problem (2) is also proved to

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program.

produce identical solution to (1) under some conditions[14]. To deal with noise, one could alternatively solve a Second Order Cone Programming (SOCP) problem:

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_1, \text{ s.t. } \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2 \leq \varepsilon, \tag{3}$$

where ε is a pre-specified error tolerance. In Sparse Representation Classification[4] (SRC), the class identity $l(\mathbf{y})$ is then assigned as

$$l(\mathbf{y}) = \underset{j\in\{1,\cdots,C\}}{\operatorname{argmin}} \ r_j(\mathbf{y}), \tag{4}$$

where $r_j(\mathbf{y}) \doteq \|\mathbf{y} - \mathbf{X}\delta_j(\boldsymbol{\beta})\|_2$ is the reconstruction residual associated with class i, C is the number of classes and the function $\delta_j(\boldsymbol{\beta})$ sets all the coefficients of $\boldsymbol{\beta}$ to 0 except those corresponding to the jth class.

Kernel Sparse Representation[6] (KSR) casts the original SR problem onto the high-dimensional feature space. Suppose there exists a feature mapping function $\Phi : \mathcal{X} \to \mathcal{F}$. It maps the feature $\mathbf{y}$ and basis $\mathbf{X}$ as

$$\begin{aligned} \mathbf{y} \to & \Phi(\mathbf{y}) \\ \mathbf{X} \to & \mathbf{U} = [\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), \cdots, \Phi(\mathbf{x}_N)]. \end{aligned} \tag{5}$$

The problem (3) is then converted to

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_1, \text{ s.t. } \|\mathbf{U}\boldsymbol{\beta} - \Phi(\mathbf{y})\|_2 \leq \varepsilon,$$

Or equivalently, the non-constraint form

$$\min_{\boldsymbol{\beta}} \|\mathbf{U}\boldsymbol{\beta} - \Phi(\mathbf{y})\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$

Gao *et al.* [6] argue that the sparse representation still maintains in the space $\mathcal{F}$ and could be solved more effectively. Their experimental results support this assumption.

It is important to notice that we do not pay attention to the dictionary learning, which is considered as another part of sparse representation[13]. In this work, the dictionary is fixed.

B. Orthogonal matching pursuit

Before the compressed sensing theory was proposed, numerous approaches had been applied for sparse approximation[15], [10], [16]. Orthogonal Matching Pursuit (OMP) is one of the approaches. Tropp and Gilbert[11] proved OMP's recoverability and showed its remarkable efficiency. In specific, the OMP solves the following problem in a greedy fashion.

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_0, \text{ s.t. } \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2 \leq \varepsilon, \tag{6}$$

Given a dictionary $\mathbf{X} \in \mathbb{R}^{d\times N}$, the computational complexity of linear programming is in $O(d^2N^{\frac{3}{2}})$, while OMP is in $O(dN)$[11]. The paradigm of OMP is shown in Algorithm 13.

Inspired by the simplicity and efficiency of OMP, in the following section, we try to extend it to the high-dimensional feature space defined by kernel functions.

Algorithm 1: Orthogonal Matching Pursuit

```
Input:
  • A normalized observation $\mathbf{y} \in \mathbb{R}^d$.
  • A dictionary $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N] \in \mathbb{R}^{d\times N}$.
  • A recovery residual $0 < \varepsilon \ll 1$.
  • A sparsity upper bound $0 < \eta < N$.
1  begin
2      Initialize the residual $\mathbf{r}_0 = \mathbf{y}$, index set $\Lambda_0 = \varnothing$ and
       selected basis set $\Omega_0 = \varnothing$;
3      for $t \leftarrow 1$ to $\eta$ do
4          $\lambda_t = \operatorname{argmax}_{i=1,\dots,N} \langle r_{t-1}, \mathbf{x}_i \rangle$;
5          $\Lambda_t = [\Lambda_{t-1} \ \ \lambda_t]$ ;
6          $\Omega_t = [\Omega_{t-1} \ \ \mathbf{x}_{\lambda_t}]$;
7          Solve the least-squares problem:
8              $\boldsymbol{\beta}_t = \operatorname{argmin}_{\boldsymbol{\beta}} \|\Omega_t\boldsymbol{\beta} - \mathbf{y}\|_2$;
9          Calculate the new residual:
10             $\mathbf{r}_t = \mathbf{y} - \Omega_t\boldsymbol{\beta}_t$ ;
11         if $\|\mathbf{r}_t\|_2 < \varepsilon$ then break;
12     Retrieve signal $\boldsymbol{\beta}$ according to $\boldsymbol{\beta}_t$ and $\Lambda_t$;
13 end
Output:
  • SR coefficient $\boldsymbol{\beta} \in \mathbb{R}^N$.
```

III. Kernel OMP: orthogonal matching pursuit in a high-dimensional space

Theorem 3.1. *All the steps in OMP could be expressed in the form of inner products between* $\mathbf{y}$ *and* $\mathbf{x}_i$, $i = 1, 2, \cdots, N$.

Proof:

For the calculation of least square problem (step 7), it is well known that the solution is

$$\boldsymbol{\beta}_t = \left(\Omega_t^\top \Omega_t\right)^{-1} \Omega_t^\top \mathbf{y}, \tag{7}$$

where Ω_t is a subset of $\mathbf{X}$ thus the above equation could be expressed in the form of inner products between $\mathbf{y}$ and $\mathbf{x}_i$.

Secondly, for step 11, the ℓ_2 norm of residual could be obtained as

$$\begin{aligned} \|\mathbf{r}_t\|_2^2 =& (\mathbf{y} - \Omega_t\boldsymbol{\beta}_t)^\top (\mathbf{y} - \Omega_t\boldsymbol{\beta}_t) \\ =& \mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\Omega_t\boldsymbol{\beta}_t + \boldsymbol{\beta}_t^\top\Omega_t^\top\Omega_t\boldsymbol{\beta}_t, \end{aligned} \tag{8}$$

where all the terms are in the form of inner product.

Finally, the calculation of residual vector $\mathbf{r}$ (step 10) is unnecessary, considering that we only use it for computing the correlations (step 4) between $\mathbf{r}$ and $\mathbf{x}_i, \forall i$. Instead, one can get the correlations by substituting $\mathbf{r}_t = \mathbf{y} - \Omega_t\boldsymbol{\beta}_t$ into step 4 which yields the new expression of the correlation

$$\begin{aligned} \langle r_{t-1}, \mathbf{x}_i \rangle =& \langle \mathbf{y} - \Omega_{t-1}\boldsymbol{\beta}_{t-1}, \mathbf{x}_i \rangle \\ =& \mathbf{y}^\top\mathbf{x}_i - \boldsymbol{\beta}_{t-1}^\top\Omega_{t-1}^\top\mathbf{x}_i, \end{aligned} \tag{9}$$

which is still the combination of inner products. ■

Inspired by this interesting finding, we could play the kernel trick easily for OMP. Given the kernel function that determines the high-dimensional space $\mathcal{F}$

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle. \tag{10}$$

Accordingly, the Gram matrix of set $\mathbf{U} = \Phi(\mathbf{X})$ in space $\mathcal{F}$ writes

$$\Psi_{i,j} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \tag{11}$$

where $\Psi \in \mathbb{R}^{N \times N}$. Let $\boldsymbol{\kappa} \in \mathbb{R}^N$ denote the inner product vector between $\Phi(\mathbf{y})$ and $\Phi(\mathbf{x}_i)$ $\forall i$, *i.e.*

$$\boldsymbol{\kappa}_i = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{y}) \rangle. \tag{12}$$

The KOMP iteratively achieve the high-dimensional SR via performing the following optimization.

$$\min_{\boldsymbol{\beta}} \ \|\boldsymbol{\beta}\|_0, \text{ s.t. } \|\mathbf{U}\boldsymbol{\beta} - \Phi(y)\|_2 \leq \varepsilon, \tag{13}$$

Based upon the analysis in Theorem 3.1, one could rewrite the OMP procedure in the fashion of kernels. The paradigm of Kernel OMP is illustrated in Algorithm 2, where we follow the manner of Matlab to denote matrix elements and the subsets.

From the paradigm we can see that KOMP only cares about formulation of the kernel function. Therefore, we can draw the conclusion that *Kernel OMP achieves sparse representation in any high-dimensional (possibly infinite) Hilbert space $\mathcal{F}$, which is implicitly determined by a proper kernel function*[1].

Compared with the FSS algorithm[9], KOMP enjoys many advantages, *e.g.* simple implementation, clear upper bound for iteration number (N) and the self-explaining parameter ε (compared with the trade-off parameter λ in [9]). We employ the KOMP algorithm to solve the SR problem and achieve superior performance as reported in the experimental part.

Algorithm 2: Kernel OMP

```
Input:
  • A kernel matrix $\Psi \in \mathbb{R}^{N \times N}$ defined in (11).
  • A kernel vector $\boldsymbol{\kappa} \in \mathbb{R}^N$ defined in (12).
  • A kernel scalar value $\theta = \Phi(\mathbf{y})^\top \Phi(\mathbf{y})$.
  • A recovery residual $0 < \varepsilon \ll 1$.
  • A sparsity upper bound $0 < \eta < N$.
1  begin
2      Initialize the index set $\Lambda_0 = \varnothing$;
3      for $t \leftarrow 1$ to $\eta$ do
4          $\lambda_t = \operatorname{argmax}_{i=1,\dots,N}(\boldsymbol{\kappa}_i - \boldsymbol{\beta}_t^\top \Psi[i, \Lambda_t])$ ;
5          $\Lambda_t = [\Lambda_{t-1} \ \lambda_t]$ ;
6          Obtain the subset of $\Psi$ and $\boldsymbol{\kappa}$:
7              $\Psi_t = \Psi[\Lambda_t, \Lambda_t], \quad \boldsymbol{\kappa}_t = \boldsymbol{\kappa}[\Lambda_t]$ ;
8          Solve the least-squares problem:
9                  $\boldsymbol{\beta}_t = \Psi_t^{-1} \boldsymbol{\kappa}_t$ ;
10         Calculate the new residual:
11                 $\|\mathbf{r}_t\|^2 = \theta - \boldsymbol{\kappa}_t^\top \boldsymbol{\beta}_t$ ;
12         if $\|\mathbf{r}_t\|^2 < \varepsilon^2$ then break;
13     Retrieve signal $\boldsymbol{\beta}$ according to $\boldsymbol{\beta}_t$ and $\Lambda_t$;
14 end
Output:
  • Kernel SR coefficient $\boldsymbol{\beta} \in \mathbb{R}^N$.
```

Please note that our algorithm is unrelated to the algorithm termed Kernel Matching Pursuit[17] where the dictionary is composed of kernel functions, rather than the implicitly mapped vectors $\Phi(\mathbf{x}_i)$ in our case.

IV. S-KOMP: NON-ITERATIVE KOMP

The conventional method of SR treats the atoms in the dictionary equally. However, in some applications, we do have additional prior knowledge for the atoms. For example, in SRC[4], each gallery face in the dictionary comes with its own label. To utilize this type of prior knowledge, various methods were proposed[18], [19], [20]. In particular, one could assume that the sparse coefficients in the same group tend to be zero or nonzero simultaneously. The assumption, termed Group-Sparisty, usually leads to higher recovery accuracy when the underlying group structure is consistent with the data[20].

Naseem *et al.* [21] proposed Linear Regression Classification (LRC) for face recognition. Given a test face $\mathbf{y}$, a series of linear regressions are conducted based on the face-sets belonging to different individuals respectively. In other words, for each linear representation, all the coefficients of one group/class is nonzero while the others are all forced to be zero. From the perspective of sparse representation, LRC could be considered as a SRC approach with a *rigid group-sparsity constraint*.

If we introduce the same constraint to KOMP, the basis of sparse representation is already selected according to their labels. Thus, the original iterative paradigm will degenerate to a single-step LS procedure. That is, given the coefficient $\mathbf{x}$ is not too sparse, a great improvement with respect to efficiency.

We term the constraint KOMP as Single-step KOMP (S-KOMP), the algorithm is illustrated in Algorithm 3. Analogous to LRC, the identity of $\mathbf{y}$ is assigned to the class with the smallest residual $\mathbf{r}_i$.

Algorithm 3: Single-step KOMP

```
Input:
  • A kernel matrix $\Psi \in \mathbb{R}^{N \times N}$ defined in (11).
  • A kernel vector $\boldsymbol{\kappa} \in \mathbb{R}^N$ defined in (12).
  • A label vector $\mathbf{l} \in \mathbb{R}^N$ corresponding to
    $[\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_2), \cdots, \Phi(\mathbf{x}_N)]$.
  • A kernel scalar value $\theta = \Phi(\mathbf{y})^\top \Phi(\mathbf{y})$.
  • A recovery residual $0 < \varepsilon \ll 1$.
  • A sparsity upper bound $0 < \eta < N$.
1  begin
2      The class number $C = max(\mathbf{l})$;
3      for $i \leftarrow 1$ to $C$ do
4          Obtain the subset of $\Psi$ and $\boldsymbol{\kappa}$:
5          $\Psi_i = \Psi[\mathbf{l} = i, \mathbf{l} = i], \quad \boldsymbol{\kappa}_i = \boldsymbol{\kappa}[\mathbf{l} = i]$ ;
6          Solve the least-squares problem:
7                  $\boldsymbol{\beta}_i = \Psi_i^{-1} \boldsymbol{\kappa}_i$ ;
8          Calculate the norm of residual:
9                  $\mathbf{r}_i = \sqrt{\theta - \boldsymbol{\kappa}_i^\top \boldsymbol{\beta}_i}$ ;
10 end
Output:
  • Kernel SR coefficients $[\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_C]$.
  • Kernel SR residual norm $[r_1, r_2, \cdots, r_C]$.
```

[1]The kernel function should satisfy the Mercers' theorem[7]

V. Experiment

A. General setting

In this section, a series of experiments is designed and conducted to verify the proposed methods. The signal-recovery capability of KOMP is tested comparing with two state-of-the-art SR solvers *i.e.* the standard SOCP solver (from the CVX package[22]) and FSS[2][9]. For face recognition, KOMP and S-KOMP are both performed comparing with SRC[4], KSRC[6] and LRC[21]. All the algorithms are implemented in Matlab on PC with a quad-core CPU and 8G memory while only one core is enabled when measuring the running speed.

B. Synthetic signal recovery

The KOMP seeks sparse representation on the high-dimensional feature space determined by certain kernel function $\mathbf{K}(\mathbf{x}, \mathbf{y})$. To justify the recovery power of KOMP, one needs to generate dictionary $\mathbf{U} = [\Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2), \cdots, \Phi(\mathbf{x}_N)]$ and the target signal $\boldsymbol{\beta}$. Without explicit Φ, it is impossible to generate high-dimensional feature $\Phi(\mathbf{x})$ from the original one $\mathbf{x}$. Therefore, we directly generate the mapped atoms $\Phi(\mathbf{x}_i)$, $\forall i$ and $\boldsymbol{\beta}$ and assume they all belong to normal distribution. The observation, or the estimation target is then calculated as

$$\Phi(y) = \mathbf{U}\boldsymbol{\beta} \tag{14}$$

Given the randomly generated dictionary $\mathbf{U}$ and $\Phi(\mathbf{y})$, the standard SOCP solver, KOMP and FSS are performed to estimate the sparse signal. Since in signal recovery tasks no label information is provided, we don't conduct S-KOMP in this scenario. In our setting, the dictionary $\mathbf{U} \in \mathbb{R}^{125\times500}$ and signal $\boldsymbol{\beta} \in \mathbb{R}^{500}$ are generated so that their entries belonging to the distribution $\mathcal{N}(0, 0.5)$. As to the parameter selection, for CVX and FSS, we use $\lambda = 1e-5$ here, which follows the setting in [6]; for KOMP, we set $\varepsilon = 0.001$.

Figure 1 illustrates a demo of the signal recovery trail. Figure 1(a) shows the original signal $\boldsymbol{\beta}$ with sparsity 20, *i.e.* 20 of the signal elements are nonzero. Figure 1(b), Figure 1(c) and Figure 1(d) show the recovery results for CVX, FSS and KOMP respectively. From the figures, we can see that CVX and FSS illustrate similar recovery performances which are both inferior to KOMP.

The trail is repeated for 50 times, with different sparsity levels of the signal. The averaged running time and the recovery error (with their standard deviations) are reported in Table I.

As illustrated in the table, KOMP is always the fastest solver and the superiority is up to 850% (FSS *vs.* KOMP, S-10). As regards the recovery error, KOMP performs best for more sparse signals while CVX performs well for less sparse ones. The FSS algorithm, on the other hand, illustrates the lowest efficiency and recovering capability on the synthetic data.

[2]We obtain the code from H. Lee's homepage and keep it unchanged during our implementation.

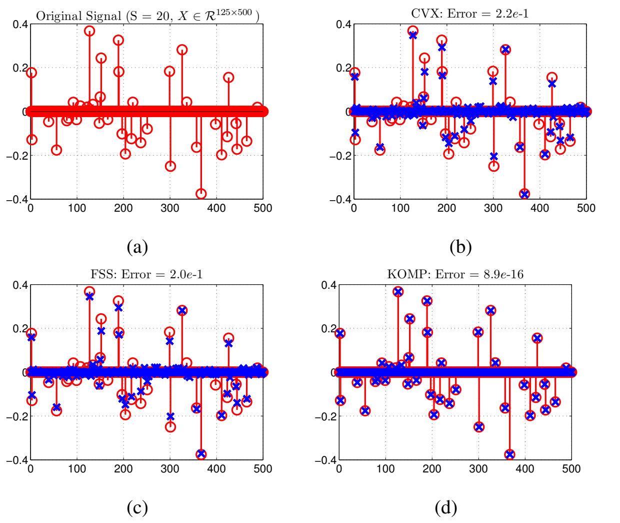

Fig. 1. Signal recovery comparison. The horizontal axis represents the signal dimensions while the vertical axis corresponds to the signal value. The red stems indicates the original signal in all the plots while the blue crossing symbols stand for the estimated one.

		S-5	S-10	S-20	S-50	S-100
Recovery error	CVX	0.14 ± 0.09	0.24 ± 0.10	0.34 ± 0.09	$\mathbf{0.57 \pm 0.09}$	$\mathbf{0.82 \pm 0.08}$
	FSS	0.13 ± 0.08	0.20 ± 0.08	$\mathbf{0.31 \pm 0.11}$	0.76 ± 0.16	1.09 ± 0.11
	KOMP	$\mathbf{0.04 \pm 0.06}$	$\mathbf{0.14 \pm 0.10}$	0.33 ± 0.17	0.89 ± 0.17	1.24 ± 0.12
Running time	CVX	1.21 ± 0.11	1.16 ± 0.05	1.14 ± 0.04	1.11 ± 0.04	1.12 ± 0.04
	FSS	1.85 ± 0.79	1.89 ± 0.04	1.84 ± 0.30	1.66 ± 0.11	1.69 ± 0.11
	KOMP	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.03 \pm 0.01}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.01}$

TABLE I
The comparison of recovery error and recovery speed (s). "S-$*$" denotes the sparsity value is $*$. Note that the recovery error r is obtained via $r = \|\mathbf{x} - \mathbf{x}^\star\|_2 / \|\mathbf{x}\|_2$, where $\mathbf{x}^\star$ is the estimated signal. Each best performance with certain sparsity is shown in bold type.

C. Face recognition

A well-known application of SR is face recognition[4], [12], [6]. We hereby compare KOMP, S-KOMP, SRC, KSRC (with FSS) and LRC by solving the face recognition problem on the datasets Yale-B[23]. Yale-B dataset contains $2,414$ well-aligned face images from 38 individuals under various lighting conditions, as illustrated in Figure 2. For each subject, we randomly choose 30 images to compose the gallery and other 30 images for testing.

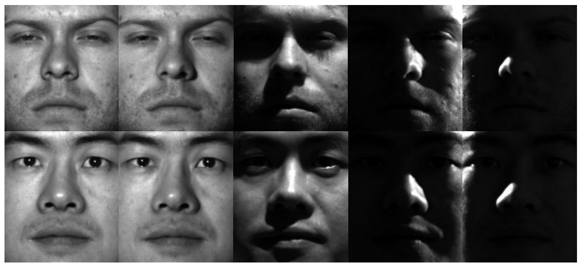

Fig. 2. The demonstration of Yale-B dataset with extreme illumination conditions.

To define the high-dimensional space, we adopt the RBF kernel for all the kernel related algorithm, *a.k.a.* KSRC, KOMP and S-KOMP. The RBF kernel writes

$$\mathbf{K}(x, y) = \exp(-\frac{\|x - y\|^2}{\sigma}) \quad (15)$$

The parameter σ is fixed at 32 for both KOMP and S-KOMP. We follow the parameter setting in the paper [6] and [4] for KSRC and SRC respectively. The residual threshold in KOMP is set as $\varepsilon = 0.001$. The dimensionality of the faces are reduced to certain values from $\{25, 50, 100, 200\}$, via PCA projection (Eigenface) or random projection (Randomface). The recognition accuracies are illustrated in Table II.

		D-25*	D-50	D-100	D-200
Eigen	LRC	60.5 ± 3.0	92.3 ± 0.5	93.6 ± 0.8	94.1 ± 0.6
	SRC	80.4 ± 1.6	89.1 ± 0.9	92.8 ± 0.8	94.2 ± 0.7
	KSRC	86.6 ± 0.9	92.2 ± 0.5	$\mathbf{94.4 \pm 0.4}$	$\mathbf{95.6 \pm 0.7}$
	KOMP	85.3 ± 1.5	91.4 ± 1.2	93.3 ± 0.5	94.8 ± 0.8
	S-KOMP	$\mathbf{89.9 \pm 1.3}$	$\mathbf{92.7 \pm 0.7}$	93.4 ± 0.6	93.8 ± 0.6
Random	LRC	31.6 ± 4.4	88.5 ± 1.0	94.0 ± 1.1	94.9 ± 0.7
	SRC	80.1 ± 1.6	90.7 ± 1.0	94.7 ± 0.5	96.3 ± 0.7
	KSRC	82.3 ± 1.2	91.5 ± 0.7	$\mathbf{95.7 \pm 0.8}$	$\mathbf{96.5 \pm 0.6}$
	KOMP	79.2 ± 1.5	$\mathbf{91.7 \pm 1.4}$	95.6 ± 0.5	96.2 ± 0.9
	S-KOMP	$\mathbf{85.8 \pm 1.2}$	91.6 ± 0.5	93.9 ± 0.6	94.8 ± 0.5

TABLE II
THE COMPARISON OF ACCURACY ON YALE-B. THE BEST PERFORMANCES ARE SHOWN IN BOLD. THE $\star$ SYMBOL INDICATES THAT THE DIMENSION IS TOO LOW TO ACHIEVE THE STABLE SOLUTION FOR LRC. PLEASE REFER TO [21] FOR DETAILS.

From the table we can see that all the best performances are obtained by kernel-based methods. In particular, KOMP achieves comparable accuracy compared with KSRC. The performance gap between these two algorithms is usually less than 1.5% and KOMP even outperforms KSRC on D-100 Randomfaces. On the other hand, S-KOMP illustrate high-accuracy on low-dimensional extracted features. It beats all the other competitors on D-25 faces and D-50 Eigenfaces. In a nutshell, our methods are comparable to KSRC and even superior when the dimensionality is low.

To illustrate the excellence of our methods in terms of efficiency, we hereby exam the running speed of all the compared algorithms. We reuse the parameters selected above, except where otherwise stated.

The running speed of kernel-based methods is significantly influenced by the kernel parameter. To understand this, just imagine we set σ in (15) very large, then the elements in the kernel matrix Ψ and kernel vector $\boldsymbol{\kappa}$ will all be very close to 1. In this scenario, it is trivial to prove that for greedy method like KOMP and KSRC (FSS), the iterative procedure will converge in few steps. Thus an apparently "high" efficiency will be observed. For fair comparison, we compare the involved algorithms with the same kernel type (RBF) and same parameters $\sigma \in \{0.5, 1, 2, 4, 8, 16, 32, 64, 128\}$.

Figure 3 illustrates the running-time curves (in ms) of the compared algorithms, with different kernel parameter σ. The results are yielded using D-200 Eigenfaces. According to the plots, LRC is ranked first in terms of running speed, with overwhelming superiority. One can perform LRC on a D-200 Eigenface within 2 ms. The second fastest one is S-KMOP with the speed of around 20 ms per face. The running-time of KOMP converges from $1,000$ ms to 110 ms after σ reaches 16. SRC requires around $7,000$ ms for processing a face. As

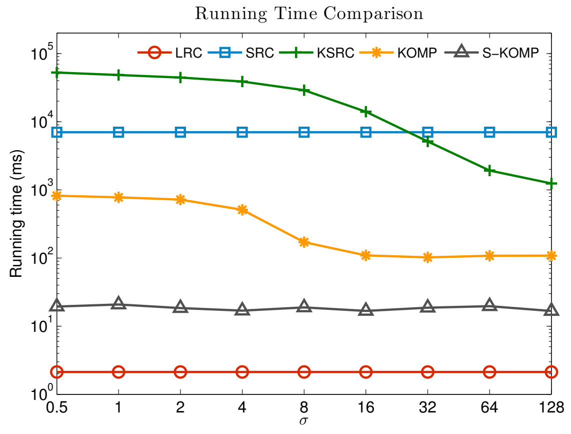

Fig. 3. Running time comparison. The results are obtained by performing the algorithms on D-200 Eigenfaces. Note that the non-kernel algorithms (SRC and LRC) are not affected by the change of σ. Thus their plots are shown as horizontal lines.

to KSRC, which employs the FSS algorithm as its solver, illustrates the lowest efficiency (up to $55,000$ ms per image) when $\sigma < 32$ and the second lowest efficiency otherwise. In the extreme case ($\sigma = 0.5$), our S-KOMP is $2,750$ times faster than KSRC, thanks to the non-iterative framework of S-KOMP. Furthermore, as we analyzed before, the elapsed time of the greedy methods (KSRC and KOMP) consistently decrease as σ grows.

Another advantage of the proposed methods is that the efficiency is insensitive to the feature dimensionality (see Figure 4). It is mainly because the size of kernel matrix Ψ and kernel vector $\boldsymbol{\kappa}$ is only related to N, *a.k.a.* the number of training samples. In contrast, KSRC does becomes slower when the dimensionality grows, even though it also performs on Ψ and $\boldsymbol{\kappa}$. The difference may be caused by the different bounds with respect to iteration-number (N in KOMP *vs.* "finite" in KSRC[9]) while the exact reason remains unclear.

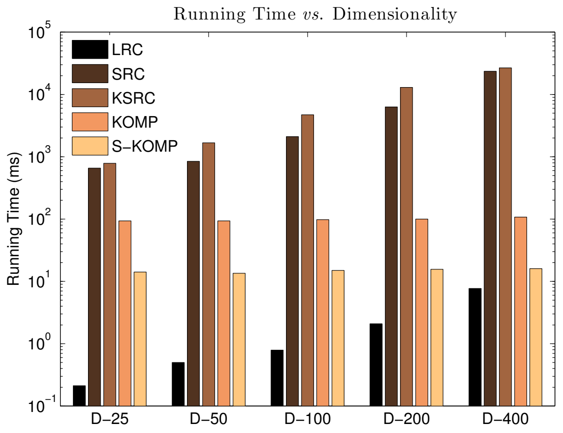

Fig. 4. Running time *vs.* dimensionality. We set $\sigma = 16$ for all the kernel related methods.

Consequently, we can draw the conclusion that KOMP and

S-KOMP are significantly faster than KSRC and SRC. In addition, the proposed algorithms can be conducted very fast even with very high-dimensional extracted features.

VI. Conclusion and future work

In summary, we have proposed two kernel-based algorithms termed KOMP and S-KOMP. The novel methods could solve the SR problem on high-dimensional Hilbert space, at a very fast speed. The experimental comparisons show that our approaches are comparable to the state-of-the-art methods in terms of accuracy. In particular, KOMP illustrates the best recovery capability on respectively sparse signals. S-KOMP achieves the highest recognition rate of 89.9% when the feature dimensionality is extremely low (D-25). More importantly, remarkable improvement in efficiency is observed for both KOMP and S-KOMP. In particular, thank to the constraint of group-sparsity, the S-KOMP algorithm is up to $2,750$ times faster than KSRC, only with marginal performance drop.

As regards future topics, we are interested in employing more advanced kernel types which treat different dimensions of the original feature discriminatively. This way, one can overcome the occlusion problem in face recognition. Another interesting direction is tuning the kernel parameter in a more sophisticated way. Higher performance is likely to be obtained based upon the modifications.

References

[1] C. Wang, S. Yan, L. Zhang, and H.J. Zhang, "Multi-label sparse coding for automatic image annotation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009, pp. 1643–1650.

[2] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, 2006.

[3] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 53–69, 2008.

[4] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, pp. 210–227, 2009.

[5] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 1794–1801, 2009.

[6] S. Gao, I. Tsang, and L.T. Chia, "Kernel sparse representation for image classification and face recognition," *Proc. Eur. Conf. Comp. Vis.*, pp. 1–14, 2010.

[7] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*, Cambridge Univ Press, 2004.

[8] S.S. Chen, D.L. Donoho, and M.A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1999.

[9] H. Lee, A. Battle, R. Raina, and A.Y. Ng, "Efficient sparse coding algorithms," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, pp. 801, 2007.

[10] Y. C. Pati, R. Rezaiifar, Y. C. Pati R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, 1993, pp. 40–44.

[11] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, pp. 4655–4666, 2007.

[12] Q. Shi, H. Li, and C. Shen, "Rapid face recognition using hashing," *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 2753–2760, 2010.

[13] M. Aharon, M. Elad, and A. Bruckstein, "k-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[14] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, pp. 1207–1223, 2006.

[15] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, pp. 3397–3415, 1993.

[16] G. Davis, S. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions with matching pursuits," *Optical Eng.*, vol. 33, 1994.

[17] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Mach. Learn.*, vol. 48, no. 1, pp. 165–187, 2002.

[18] F.R. Bach, "Consistency of the group lasso and multiple kernel learning," *J. Mach. Learn. Res.*, vol. 9, pp. 1179–1225, 2008.

[19] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 417–424.

[20] J. Huang and T. Zhang, "The benefit of group sparsity," *Arxiv preprint arXiv:0901.2962*, 2009.

[21] I. Naseem, R. Togneri, and M. Bennamoun, "Linear regression for face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 2106–2112, 2010.

[22] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming," *Available httpstanford edu boydcvx*, 2008.

[23] K.C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, 2005.