International Journal of Innovative Computing, Information and Control
Volume **7**, Number **10**, October **2011**

ICIC International ©2011 ISSN 1349-4198
pp. 5645–5667

INNOVATIVE SPARSE REPRESENTATION ALGORITHMS FOR ROBUST FACE RECOGNITION

Huining Qiu[1], Duc-Son Pham[2], Svetha Venkatesh[2]
Jianhuang Lai[3] and Wanquan Liu[2]

[1]Department of Mathematics
[3]Department of Automation
Sun Yat-sen University
No. 135, Xingang Xi Road, Guangzhou 510275, P. R. China
qiuhuining@ieee.org; stsljh@mail.sysu.edu.cn

[2]Department of Computing
Curtin University
GPO Box U1987, Perth, WA 6845, Australia
dspham@ieee.org; svetha@cs.curtin.edu.au; w.liu@curtin.edu.au

Received May 2010; revised September 2010

Abstract. *In this paper, we propose two innovative and computationally efficient algorithms for robust face recognition, which extend the previous Sparse Representation-based Classification (SRC) algorithm proposed by Wright et al. (2009). The two new algorithms, which are designed for both batch and online modes, operate on matrix representation of images, as opposed to vector representation in SRC, to achieve efficiency whilst maintaining the recognition performance. We first show that, by introducing a matrix representation of images, the size of the ℓ_1-norm problem in SRC is reduced from $O(whN)$ to $O(rN)$, where $r \ll wh$ and thus higher computational efficiency can be obtained. We then show that the computational efficiency can be even enhanced with an online setting where the training images arrive incrementally by exploiting the interlacing property of eigenvalues in the inner product matrix. Finally, we demonstrate the superior computational efficiency and robust performance of the proposed algorithms in both batch and online modes, as compared with the original SRC algorithm through numerous experimental studies.*

Keywords: Sparse representation, Incremental learning, Robust face recognition

1. Introduction.

In recent years, face recognition has been substantially studied both in the academic community and industry with many significant results achieved [1, 2, 3, 4]. The target of face recognition is to build systems which can perform automatic person identification or verification, when a digital image or a video frame sequence of that person is provided. During the past two decades, a number of face recognition algorithms, as well as their modifications, have been developed. These algorithms can be typically categorized into two classes: appearance-based and model-based approaches.

In appearance-based methods, the features are the pixel intensities in a digital face image. These pixel intensities are the quantized measurements of light radiance emitted from a person along certain rays in space, and contain abundant information which can be used to determine identity from appearance. These methods include: subspace-based methods [5, 6, 7], Hidden Markov Model (HMM) methods [8], Bayesian methods [9], Support Vector Machine (SVM) methods [10], Kernel methods [11, 12, 13, 14] and multi-resolution method [15].

For model-based approaches, general shape models of human faces are introduced, such as Elastic Bunch Graph Matching (EBGM) [16], Active Appearance Model (AAM) [17] and 3D Morphable Model method [18]. These methods employ general facial shape models as representations of faces, like face bunch graph, or model with landmark points, or model of 3D face shape and texture. The image pixels are treated as low level features which need to be extracted into high level features before adapting to these models. A face image of a person is assumed to be the output of the face model corresponding to input parameters, and face recognition is transformed into a model matching problem.

In the literature of face recognition, the appearance-based methods have been extensively studied, among which only the subspace-based algorithms are reviewed here. Some of the subspace-based algorithms include: Eigenfaces and its variants [19, 20], Fisherfaces and its numerous modifications [21, 22], Laplacianfaces and its extensions [23, 24], ICA-based methods [25], NMF-based methods [26, 27], to name only a few. Most of these techniques depend on a representation of images in a vector space structure. Algorithms then adopt statistical techniques to analyze the distribution of the object image vectors, and find effective representations in a transformed vector space (feature space) according to various criteria. Once a test image is captured, the similarity between the test image and the prototype training sets is then calculated in the feature space. Face recognition in this category is in fact a learning process with optimization techniques.

So far, face recognition in controlled environments has reached its maturity with high performance. However, face recognition in less controlled or uncontrolled environments still requires further study in order to be usable in practice. Recent standardized vendor face technology tests revealed that there are still major challenges in practical face recognition applications [28, 29, 30, 31]. The main challenges are the potential large intra-subject variations in human face image appearance due to 3D head pose changes, illumination variants (including indoor/outdoor conditions), facial expressions, occlusions with other objects or accessories (e.g., sunglasses, scarfs), facial hair and aging. As these difficulties exist in face recognition, more robust face recognition algorithms are still needed.

Recently, Wright et al. [32] have developed a new face recognition framework for the robust face recognition problem. Their work is based on a newly developed compressed sensing theory, and has shown its robust performance compared with traditional face recognition techniques. Compressed sensing is a technique first developed in signal processing community for reconstructing a sparse signal by utilizing the prior knowledge of its sparsity structure [33, 34]. Classical signal reconstruction method is to minimize the ℓ_2 norm, which is equivalent to minimizing the amount of energy in the system. The compressed sensing theory resolves to minimize the ℓ_0 norm, which is equivalently relaxed to minimizing the ℓ_1 norm under certain conditions. It yields attractive solutions which show their robust property against noises in many problems. Due to its mathematics foundation and effective framework, compressed sensing has already drawn immense attention in areas of mathematics, optimization, information theory, statistical signal processing, high-dimensional data analysis [35, 36, 37]. A survey about compressed sensing and its broad applications can be found in [38].

The idea of compressed sensing is adopted by Wright et al. in their new algorithm for face recognition, namely Sparse Representation-based Classification (SRC) algorithm. This algorithm is robust in a sense that the sparse representation is less sensitive to outliers in the face images such as occlusion or random pixel corruptions. However, the major disadvantage of the SRC algorithm is its very expensive computational cost, which limits its current applicability. Due to its vector representation, the SRC needs to solve an ℓ_1-regularized optimization problem whose size is the total number of pixels of the images, which can be extremely large when high resolution images are used.

To overcome the computational issue, two approaches can be considered. First, dimensionality reduction can be performed on the input images and then extracted feature vectors can be used instead of original pixel features. However, this approach might potentially lose beneficial information in the original images. Second, equivalent reformulation of the problem can be pursued to find the solution faster, and the optimization problem can be solved with proper acceleration technique. In this paper, we follow the second alternative and aim to reduce computational complexity of the SRC algorithm by working directly on 2D representation of images. Our method is accomplished by reformulating the 2D images sparse representation problem, and then solving it through the existing ℓ_1 optimization techniques. For convenience, hereinafter, we name the original SRC algorithm as 1D-SRC, and our new algorithm as 2D-SRC. Further, we consider applying the 2D-SRC algorithm in an incremental learning context, and propose an incremental 2D-SRC learning procedure which has proved to be more efficient.

This paper is organized as follows: in Section 2, we review the SRC algorithm and propose our 2D extension algorithm and the incremental computing procedure, followed by the complexity analysis; then, in Section 3, we compare the performance of 1D-SRC with 2D-SRC and the incremental algorithms on some benchmark datasets, and reveal that the proposed algorithms can speed up without decreasing the recognition performance; finally, we conclude the paper in Section 4.

2. SRC Algorithms and Their Computational Complexity.

We first present the original 1D-SRC algorithm and analyze its high computational cost problem, then we propose an equivalent formulation and induce a 2D-SRC algorithm which can be solved much faster. After that, we further extend the 2D-SRC algorithm into the incremental learning context. At last, we discuss the computational complexity of the three algorithms.

2.1. The SRC algorithm.

In this subsection, we briefly review the sparse representation face model and the SRC algorithm.

In face recognition research, it is generally conceived that there exists a face subspace which is formed by one person's face images under different variations (e.g., pose, illumination, expression). As a result, linear models can be used to approximate these "face subspaces". The recently proposed sparse representation-based face model is developed based on this hypothesis, and it uses all known training sample images to span a face subspace. For a test face image whose class label is unknown, one tries to reconstruct the test image sparsely from the training samples.

The motivation of this model is that if given sufficient training samples of one person, then any new (test) sample for this person will approximately lie in the linear span of the training samples associated with this person. To be more precise, let us say, a database consists of k classes denoted as:

$$\mathcal{A} = \{\mathbf{A}_{1,1}, \ldots, \mathbf{A}_{1,n_1}, \ldots, \mathbf{A}_{k,1}, \ldots, \mathbf{A}_{k,n_k}\}$$

where $\mathbf{A}_{i,l}$ is the l-th image belonging to class i, n_i is number of samples for class i, $i = 1, \ldots, k$. By stacking pixels of each image $\mathbf{A}_{i,l}$ into a column vector $\mathbf{v}_{i,l}$, one can build up a matrix A to represent the training samples

$$A = [\mathbf{v}_{1,1}, \ldots, \mathbf{v}_{1,n_1}, \ldots, \mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,n_k}] \in \mathcal{R}^{L\times N} \tag{1}$$

where $L = hw$ is the number of pixels for a $h \times w$ image, $N = n_1 + \ldots + n_k$ is the total number of samples for all classes.

For a new test image $\mathbf{y}$, we then can represent it using linear combination of samples from the database

$$\mathbf{y} = A\mathbf{x}_0. \tag{2}$$

Most ideally, if $\mathbf{y}$ is from person i, then based on the assumption that person i's face subspace is sufficient to represent itself, so the coefficients $\mathbf{x}_0$ should have a form of

$$\mathbf{x}_0 = [0, \ldots, 0, \alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,n_i}, 0 \ldots, 0] \tag{3}$$

in other words, the solution $\mathbf{x}_0$ in linear Equation (2) should only have non-zero values at positions corresponding to the same person as the test image, therefore, it should be very sparse. Thus, one can use "sparsity" as a heuristic principle for solving the linear Equation (2), even though not knowing the true identity of the test image. For this purpose, one can set up an objective to measure the "sparsity" of the coefficients $\mathbf{x}$. From the compressed sensing theory one knows that a restriction of ℓ_1-norm has an effect of producing sparse solutions. So, this leads to the following least ℓ_1-norm problems:

$$(\ell_1): \quad \begin{array}{l} \hat{\mathbf{x}}_1 = \arg\min\limits_{\mathbf{x}} \|\mathbf{x}\|_1 \\ \text{s.t.} \;\; A\mathbf{x} = \mathbf{y}. \end{array} \tag{4}$$

and

$$(\ell_1^s): \quad \begin{array}{l} \hat{\mathbf{x}}_1 = \arg\min\limits_{\mathbf{x}} \|\mathbf{x}\|_1 \\ \text{s.t.} \;\; \|A\mathbf{x} - \mathbf{y}\|_2 \le \epsilon. \end{array} \tag{5}$$

The above two models are both used in the SRC algorithm, and they are different because (4) is a noise-free model and (5) is a model in the existence of noises. However, they can be solved using the same optimization technique.

Now the SRC algorithm can be summarized as in Algorithm 1:

Algorithm 1 Sparse Representation-based Classification (1D-SRC by Wright et al. [32])

```
1: Input: a matrix of training samples for $k$ classes
       $A = [\mathbf{v}_{1,1}, \ldots, \mathbf{v}_{1,n_1}, \ldots, \mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,n_k}] \in \mathcal{R}^{(wh)\times N}$
   (each column of $A$ is a vectorization of training sample image $\mathbf{A}_{i,i_l}$); the class labels
   $class(p)$, $p = 1, \ldots, k$; the corresponding class labels $label(i)$ of each training sample
   vector $A(:, i)$; a test sample $\mathbf{y} \in \mathcal{R}^{(wh)\times 1}$; an optional error tolerance parameter $\epsilon > 0$.
2: Normalize the columns of $A$ to have unit $\ell^2$-norm.
3: Solve the $\ell_1$-norm minimization problem
       $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \;\; \text{s.t.} \;\; A\mathbf{x} = \mathbf{y}.$
   or alternatively solve the $\ell_1$-norm minimization problem
       $\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \;\; \text{s.t.} \;\; \|A\mathbf{x} - \mathbf{y}\|_2 \le \epsilon.$
4: Compute the per-class residuals
       $r_p(\mathbf{y}) = \|\mathbf{y} - A\,\delta_p(\mathbf{x})\|_2 \;\; \text{for} \;\; p = 1, \ldots, k.$
   where $\delta_p(\mathbf{x})$, for $p = 1, \ldots, k$ is a new vector for the $p$-th class whose entries are
   defined as:
       for $i = 1, \ldots, N$,  $\delta_p^{(i)}(\mathbf{x}) = \begin{cases} x_i, & \text{if } label(i) \text{ is } class(p) \\ 0, & \text{otherwise.} \end{cases}$
5: Output: $identity(\mathbf{y}) = class(p_*)$, $p_* = \arg\min_p r_p(\mathbf{y})$.
```

Based on the experimental results reported in [32], the SRC algorithm demonstrates robust performance for face datasets with noises and occlusions, and can attain high recognition rates.

2.2. **The 2D-SRC algorithm.** In this section, we are concerned with the computational cost of the SRC algorithm. From Algorithm 1, we can find that the most time consuming step of the SRC algorithm is Step 3, which needs to solve the ℓ_1-norm minimization problem (4) or (5). Further, the problem size of the ℓ_1-norm minimization is determined by the size of matrix A. More precisely, it is proportionally increasing to $size(A) = L \times N$, where L is the image dimensionality, i.e., the total pixel counts $L = wh$, and N is the training sample size. So, if either the image dimensionality or the training sample size is very large, then the computation of solving the ℓ_1-norm minimization problem will become very slow, or even failed. For example, a typical face database usually contains images whose facial area can easily take up $90 \times 90 = 8100$ pixels, which is an extremely high dimensionality and can cause great computational costs. In this case, the SRC algorithm does not work well because it cannot find the least ℓ_1-norm solution effectively. To overcome this problem, the authors in [32] proposed to downsize the original images into smaller images which can make the computation possible, and all their computations there are executed upon these smaller images. Although such downsampling of the images can reduce the problem size and enable to compute attractable solutions, it also leads to loss of discriminant information which are important for classification. Though there exist other dimensionality reduction methods which can reduce the problem size of ℓ_1-norm minimization, it is better to preserve the discriminant information by using the original images directly. Next, we devote to developing a 2D-SRC algorithm for this purpose.

We first analyze the 1D-SRC model before investigation of a 2D-SRC model. According to Fuchs [39, 40] and Boyd [41], the linear programming problem (4) and the quadratic programming problem (5) can be equivalently transformed into the following ℓ_1-norm regularization problem

$$\hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \|A\mathbf{x} - \mathbf{y}\|_2^2 \quad \text{for some } \lambda. \tag{6}$$

This transformation is equivalent provided the parameter λ is adequately chosen. Thus this transformation can unify problems (4) and (5) with an aim of simplifying the process of the SRC algorithm. We start inducing a 2D-SRC model on basis of this new transformation.

We notice that in (6) the matrix A is formed by stacking each $h \times w$ image $\mathbf{A}_i$ into a long vector $\mathbf{v}_i$ and then putting it to be a column of the matrix, thus A is a large matrix since its row number is equal to the dimensionality of images. In order to overcome the dimensionality curse problem, we represent each image as a matrix instead of a vector, to derive a 2D sparse representation model that is similar to (6). For convenience, hereafter we use the bold symbols $\mathbf{A}_i$ and $\mathbf{A}$ to denote the matrices of the training images and the test image respectively. For a test image $\mathbf{A}$, we suppose it can be represented by the whole training images with a linear combination as follows:

$$\mathbf{A} = \sum_{i=1}^{N} x_i \mathbf{A}_i \tag{7}$$

where x_i $(i = 1, \ldots, N)$ are the coefficients need to be determined. Further, we can formulate the following ℓ_1-norm regularization problem under the sparse representation assumption

$$\hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \left\|\mathbf{A} - \sum_{i=1}^{N} x_i \mathbf{A}_i\right\|_F^2 \tag{8}$$

where $\mathbf{A}$ is an $h \times w$ image matrix whose label is to be determined, $\mathbf{A}_i$ $(i = 1, \dots, N)$ is a set of training image matrices whose class labels are already known, $\| \cdot \|_F$ is the Frobenius matrix norm, $\mathbf{x} = [x_1, \dots, x_N]^T$ is the sparse coefficients, and λ is a regularization parameter that needs to be adequately chosen. Denote $\mathcal{K}^{h \times w}$ the vector space containing all real matrices with h rows and w columns, we can introduce the Frobenius inner product as follows:

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \sum_i \sum_j A_{ij} B_{ij}. \tag{9}$$

The Frobenius inner product is related to the Frobenius matrix norm by the following equation

$$\langle \mathbf{A}, \mathbf{A} \rangle_F = \sum_i \sum_j A_{ij}^2 = \|\mathbf{A}\|_F^2. \tag{10}$$

Accordingly, we can rewrite the reconstruction error term in 2D-SRC model (8) as:

$$\begin{aligned}
& \|\mathbf{A} - \sum_i x_i \mathbf{A}_i\|_F^2 & (11)\\
= & \left\langle \mathbf{A} - \sum_i x_i \mathbf{A}_i, \mathbf{A} - \sum_i x_i \mathbf{A}_i \right\rangle_F & (12)\\
= & \sum_i \sum_j x_i x_j \langle \mathbf{A}_i, \mathbf{A}_j \rangle_F - 2 \sum_i x_i \langle \mathbf{A}_i, \mathbf{A} \rangle_F + \|\mathbf{A}\|_F^2 & \\
= & \mathbf{x}^T \mathbf{Q} \mathbf{x} - 2\mathbf{b}^T \mathbf{x} + c & (13)
\end{aligned}$$

where

$$\mathbf{Q} = \begin{pmatrix} \langle \mathbf{A}_1, \mathbf{A}_1 \rangle_F & \dots & \langle \mathbf{A}_1, \mathbf{A}_N \rangle_F \\ \vdots & \ddots & \vdots \\ \langle \mathbf{A}_N, \mathbf{A}_1 \rangle_F & \cdots & \langle \mathbf{A}_N, \mathbf{A}_N \rangle_F \end{pmatrix} \in \Re^{N \times N}, \tag{14}$$

$$\mathbf{b} = \begin{pmatrix} \langle \mathbf{A}_1, \mathbf{A} \rangle_F \\ \langle \mathbf{A}_2, \mathbf{A} \rangle_F \\ \vdots \\ \langle \mathbf{A}_N, \mathbf{A} \rangle_F \end{pmatrix} \in \Re^{N \times 1}, \tag{15}$$

$$c = \|A\|_F^2. \tag{16}$$

As we can see from (13) that the 2D-SRC model objective function is actually a quadratic form, and the problem size is only proportional to the number of input samples N. As $\mathbf{Q}$ is symmetric and positive semidefinite, we can always find a matrix with full column rank $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_r]$ such that

$$\mathbf{Q} = \mathbf{P}^T \mathbf{P}. \tag{17}$$

Then we can rewrite the quadratic form (13) as:

$$(13) = (\mathbf{P}\mathbf{x})^T (\mathbf{P}\mathbf{x}) - 2\mathbf{b}^T \mathbf{x} + c \tag{18}$$

if further we can find a vector $\mathbf{z}$ such that

$$\mathbf{P}^T \mathbf{z} = \mathbf{b} \tag{19}$$

then (18) becomes

$$\begin{aligned}(18) &= (\mathbf{Px})^T(\mathbf{Px}) - 2\left(\mathbf{P}^T\mathbf{z}\right)^T\mathbf{x} + c && (20)\\ &= (\mathbf{Px})^T(\mathbf{Px}) - 2\mathbf{z}^T(\mathbf{Px}) + \mathbf{z}^T\mathbf{z} - \mathbf{z}^T\mathbf{z} + c && (21)\\ &= \|\mathbf{Px}-\mathbf{z}\|^2 - \mathbf{z}^T\mathbf{z} + c. && (22)\end{aligned}$$

And finally, we are able to transform the original 2D-SRC problem equivalently to

$$\begin{aligned} & \hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \|\mathbf{A} - \sum_i x_i\mathbf{A}_i\|_F^2 && \text{for some } \lambda, && (23)\\ \Leftrightarrow\quad & \hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \left(\|\mathbf{Px}-\mathbf{z}\|^2 - \mathbf{z}^T\mathbf{z} + c\right) && \text{for some } \lambda, && (24)\\ \Leftrightarrow\quad & \hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \|\mathbf{Px}-\mathbf{z}\|^2 && \text{for some } \lambda. && (25)\end{aligned}$$

Thus, the 2D-SRC model (8) can lead to a smaller size ℓ_1-minimization problem (25). The problem size now becomes $size(\mathbf{P}) = r \times N$, and is smaller than the problem size of the 1D-SRC model, which is $L \times N$. We can prove $r \leq \min(L, N)$ (see Lemma 1 in the Appendix).

The only question left is how to find feasible $\mathbf{P}$ and $\mathbf{z}$ that satisfy (17) and (19). In order to find a P, let the compact Singular Value Decomposition (SVD) of $\mathbf{Q}$ be

$$\mathbf{Q} = \mathbf{USU}^T = \left(\mathbf{US}^{1/2}\right)\left(\mathbf{US}^{1/2}\right)^T \tag{26}$$

where S is a nonsingular diagonal matrix and U is a orthnormal matrix with full column rank. By simply letting

$$\mathbf{P} = \left(\mathbf{US}^{1/2}\right)^T \tag{27}$$

one can obtain a desirable solution for P. Further, since the columns of $\mathbf{U}$ are orthogonal, one can easily solve $\mathbf{P}^T\mathbf{z} = \mathbf{b}$ by

$$\begin{aligned} & \mathbf{P}^T\mathbf{z} = \mathbf{b} && (28)\\ \Rightarrow\quad & \mathbf{PP}^T\mathbf{z} = \mathbf{Pb} && (29)\\ \Rightarrow\quad & \left(\mathbf{US}^{1/2}\right)^T\left(\mathbf{US}^{1/2}\right)\mathbf{z} = \mathbf{Pb} && (30)\\ \Rightarrow\quad & \mathbf{z} = \mathbf{S}^{-1}\mathbf{Pb}. && (31)\end{aligned}$$

Once $\mathbf{P}$ and $\mathbf{z}$ are obtained as above, the 2D-SRC model is established. This model can still be solved by the optimization technique used in [32], but the size of the problem is much smaller.

Based on the original 1D-SRC algorithm, we can now describe the 2D-SRC algorithm as Algorithm 2.

2.3. **The incremental 2D-SRC algorithm.**

In this section, we further extend the proposed 2D-SRC algorithm to an incremental learning context. Consider scenarios when training data samples are coming incrementally, it is beneficial to reuse previous computational results rather than to compute each time from scratch. For the 2D-SRC problem, when the training image samples are coming incrementally, we devote to derive an incremental learning algorithm for classifier design. In this paper, we only consider adding one image into the training image set each time for simplicity.

Assume that the current training images set is $\mathcal{A}_n = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n\}$, and a new image $\mathbf{A}_{n+1}$ is labeled and added to form $\mathcal{A}_{n+1} = \mathcal{A}_n \cup \{\mathbf{A}_{n+1}\}$. For this new images set $\mathcal{A}_{n+1}$, the 2D-SRC algorithm can be readily applied. However, this implies that we need to do SVD on the new inner product matrix $\mathbf{Q}_{n+1} = \left[\langle \mathbf{A}_i, \mathbf{A}_j\rangle_F\right]_{i,j=1,\ldots,n+1}$. In this way, we have dropped all our previous computed results, especially the SVD results on the previous inner product matrix $\mathbf{Q}_n = \left[\langle \mathbf{A}_i, \mathbf{A}_j \rangle_F\right]_{i,j=1,\ldots,n}$. As we can observe that in the 2D-SRC algorithm, computing SVD of the Frobenius inner product matrix $\mathbf{Q}$ is a critical step. If we can reduce the cost of each SVD computation, then we can further improve the 2D-SRC efficiency with incremental learning.

Algorithm 2 2D Sparse Representation-based Classification (2D-SRC)

```
1: Input: a set of training sample images for $k$ classes
       $\{\mathbf{A}_{1,1}, \ldots, \mathbf{A}_{1,n_1}, \ldots, \mathbf{A}_{k,1}, \ldots, \mathbf{A}_{k,n_k}\} \subset \Re^{h\times w}$
   the corresponding class labels $label(i)$ of each training sample image $\mathbf{A}_i$, a test sample
   $\mathbf{A} \in \Re^{h\times w}$ and a regularization parameter $\lambda > 0$.
2: Compute the Frobenius inner product matrix $\mathbf{Q}$ and vector $\mathbf{b}$ by (15).
3: Apply SVD on $\mathbf{Q}$ to find the matrix $\mathbf{P}$ and vector $\mathbf{z}$ by (27) and (28).
4: Solve the reduced $\ell_1$-minimization (1D-SRC) problem
       $\hat{\mathbf{x}}_1 = \arg\min_{\mathbf{x}} \lambda\|\mathbf{x}\|_1 + \|\mathbf{P}\mathbf{x} - \mathbf{z}\|_2^2 \quad \text{for some } \lambda.$   (32)
5: Compute the per-class residuals
       $r_p(\mathbf{A}) = \|\mathbf{A} - \sum_i \delta_p^{(i)}(\mathbf{x})\,\mathbf{A}_i\|_F \quad \text{for } p = 1,\ldots,k.$   (33)
   where $\delta_p(\mathbf{x})$, for $p = 1,\ldots,k$ is a vector for the $p$-th class whose entries are defined as:
       $\text{for } i = 1,\ldots,N, \quad \delta_p^{(i)}(\mathbf{x}) = \begin{cases} x_i, & \text{if } label(i) \text{ is } class(p) \\ 0, & \text{otherwise.} \end{cases}$   (34)
6: Output: $identity(\mathbf{y}) = class(p_*)$, $p_* = \arg\min_p r_p(\mathbf{y})$.
```

First, we notice that for any n, $\mathbf{Q}_n$ is an inner product matrix, so it is symmetric and positive semidefinite. Therefore, computing the SVD of $\mathbf{Q}_n$ is equivalent to computing the Eigenvalue Decomposition (EVD) of $\mathbf{Q}_n$. What's more favorable, all eigenvalues of $\mathbf{Q}_n$ are non-negative (see Lemma 2 in Appendix). Consequently, we can focus on how to compute the EVD of $\mathbf{Q}_{n+1}$ from the EVD of $\mathbf{Q}_n$. Mathematically, the problem can be stated as if given the EVD

$$\mathbf{Q}_n = \mathbf{U}_n \mathbf{\Lambda}_n \mathbf{U}_n^T \tag{35}$$

and when a new sample $\mathbf{A}_{n+1}$ is added, how can we compute the EVD

$$\mathbf{Q}_{n+1} \equiv \begin{bmatrix} \mathbf{Q}_n & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} = \mathbf{U}_{n+1}\mathbf{\Lambda}_{n+1}\mathbf{U}_{n+1}^T \tag{36}$$

efficiently, where $\mathbf{b} = [b_1, b_2, \ldots, b_n]^T$, $b_i = \langle \mathbf{A}_i, \mathbf{A}_{n+1}\rangle_F$ $(i = 1, \ldots, n)$ and $c = \left\langle \mathbf{A}_{n+1}, \mathbf{A}_{n+1} \right\rangle_F$ are the incremental inner products formed with the new sample.

This is actually a mathematical problem, and the following theorem provides us the relation between the eigenvalues and eigenvectors of $\mathbf{Q}_n$, $\mathbf{Q}_{n+1}$.

Theorem 2.1. *(An Interlacing property of Eigenvalues* [42]*) Assume that the eigenvalue decomposition (EVD) of* $\mathbf{Q}_{n+1}$ *is given by*

$$\begin{aligned} \mathbf{Q}_{n+1} \equiv \begin{bmatrix} \mathbf{Q}_n & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} &= \begin{bmatrix} \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^T & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{U}_n & 0 \\ 0 & 1 \end{bmatrix} \mathbf{R}_{n+1} \begin{bmatrix} \mathbf{U}_n^T & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{37}$$

where

$$\mathbf{R}_{n+1} \equiv \begin{bmatrix} \mathbf{\Lambda}_n & \mathbf{U}_n^T\mathbf{b} \\ \mathbf{b}^T\mathbf{U}_n & c \end{bmatrix} \equiv \begin{pmatrix} \lambda_1 & & & z_1 \\ & \ddots & & \vdots \\ & & \lambda_n & z_n \\ z_1 & \cdots & z_n & c \end{pmatrix} \tag{38}$$

has the property that its eigenvalues are exactly $n+1$ *roots of the following equation w.r.t. variable* d

$$\frac{z_1^2}{d-\lambda_1} + \cdots + \frac{z_n^2}{d-\lambda_n} = d - c. \tag{39}$$

Moreover, if we let $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ *be sorted, and* $d_1 < d_2 \cdots < d_n < d_{n+1}$ *be all* $n+1$ *sorted roots of (39), then the following interlacing relation holds*

$$d_1 < \lambda_1 < d_2 < \lambda_2 < \cdots < d_n < \lambda_n < d_{n+1}, \tag{40}$$

and the eigenvector $\mathbf{v}_i$ *corresponding to its eigenvalue* d_i *can be obtained by solving the eigen-equation directly, which gives*

$$\mathbf{v}_i = \frac{1}{T_i}\left[\frac{z_1}{d_i-\lambda_1}, \cdots, \frac{z_n}{d_i-\lambda_n}, 1\right], \quad for \quad i = 1, \ldots, n+1 \tag{41}$$

(T_i *is a factor normalizing* $\mathbf{v}_i$ *to be unit norm).*

Finally, we have the EVDs

$$\mathbf{V_{n+1}} = [\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}], \quad \mathbf{D_{n+1}} = diag\left([d_1, \ldots, d_{n+1}]\right) \tag{42}$$

$$\mathbf{R}_{n+1} = \mathbf{V}_{n+1}\mathbf{D}_{n+1}\mathbf{V}_{n+1}^T \tag{43}$$

$$\mathbf{U}_{n+1} = \left(\begin{bmatrix} \mathbf{U}_n & 0 \\ 0 & 1 \end{bmatrix} \mathbf{V}_{n+1}\right), \quad \mathbf{\Lambda}_{n+1} = \mathbf{D}_{n+1} \tag{44}$$

$$\mathbf{Q}_{n+1} = \mathbf{U}_{n+1}\mathbf{\Lambda}_{n+1}\mathbf{U}_{n+1}^T. \tag{45}$$

This gives the SVD of $\mathbf{Q}_{n+1}$.

In the above theorem, without losing generality we can assume all eigenvalues are distinct and all $z_i \neq 0$, and we leave intensive discussion about repeated eigenvalues or any $z_i = 0$ for Lemma 3 in the Appendix. The properties of the matrix R_{n+1} in Theorem 2.1 are well studied in mathematics community [43, 44, 45, 46]. Based on the result in Theorem 2.1, we can design a binary search algorithm to find all $n+1$ eigenvalues of $\mathbf{Q}_{n+1}$ from (34) which is presented in Algorithm 3. Then we can compute the corresponding eigenvectors through (41)-(44). This makes the incrementally computing of the SVD possible. Eventually, we can summarize the incremental learning procedure for the 2D-SRC algorithm in Algorithm 4.

Algorithm 3 Binary search for roots of Equation (39)

```
1: Input: the current $n$ eigenvalues $\lambda_1,\ldots,\lambda_n$, the values $z_1,\ldots,z_i$ and $c$, $M = norm(Q_n)$,
   convergent tolerance $\epsilon > 0$.
2: For $i = 1,\ldots,n+1$, binary search $d_i$ in interval $[\lambda_{i-1}, \lambda_i]$ (let $\lambda_0 = -M$, $\lambda_{n+1} = M$):
     let $low = \lambda_{i-1}$, $high = \lambda_i$
     do
       $d_i = (\lambda_{i-1} + \lambda_i)$;
       $fval = (d_i - c) - \sum_{i=1}^{n} z_i^2 / (d_i - \lambda_i)$;
       if $fval > 0$ then $low = mid$, else $high = d_i$;
     while $fabs(high - low) > \epsilon$
3: Output: the new $n+1$ eigenvalues $d_1,\ldots,d_{n+1}$.
```

Algorithm 4 Incremental procedure for 2D-SRC

```
1: Assume that we have established a 2D-SRC model (by Algorithm 2) for $n$ training
   images, and now a new $(n+1)$-th training image is added.
2: Given current training sample image set $\{\mathbf{A}_1, \ldots, \mathbf{A}_n\} \subset \Re^{h\times w}$ with their label set,
   a saved copy of SVD decomposition on current Frobenius inner product matrix $\mathbf{Q}_n$
   $= \mathbf{U}_n \mathbf{\Sigma}_n \mathbf{U}_n^T$, a new labeled training sample image $\mathbf{A}_{n+1} \in \Re^{h\times w}$.
3: Compute the new formed Frobenius inner product matrix entries $\mathbf{b} = [b_1, b_2, \ldots, b_n]$
   where $b_i = \langle \mathbf{A}_i, \mathbf{A}_{n+1}\rangle_F$ $(i = 1, \ldots, n)$ and $c = \langle \mathbf{A}_{n+1}, \mathbf{A}_{n+1}\rangle_F$.
4: Compute all eigenvalues $\mathbf{\Sigma_{n+1}} = diag\,(d_1, \ldots, d_{n+1})$ by Algorithm 3.
5: Compute all eigenvectors $\mathbf{U_{n+1}}$ by (41) – (44).
6: This establishes a 2D-SRC model (Algorithm 2) with training image set
   $\{\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{A}_{n+1}\}$.
7: $n \leftarrow n + 1$, and turn to Step 2.
```

2.4. **Complexity analysis.** One can see from the Algorithm 2 that the complexity of establishing a 2D-SRC model consists of two parts, the transformation step for finding parameters P and z and then solving the reduced ℓ_1-minimization problem. In fact, the main computational cost in recognition is in the optimization process for solving the ℓ_1-minimization problem. One can observe that problem size of the ℓ_1-minimization problem in 1D-SRC is $L \times N$ and the one for the 2D-SRC is $r \times N$. In most of applications, we have $L \gg N$ and $r < N$, and this indicates that the 2D-SRC algorithm can speed up significantly.

It is noted importantly that our analysis is based on an assumption that the smaller the input size of ℓ_1-norm solver, the faster it runs. Under this assumption, we can reduce the computational complexity from $O(LN)$ to $O(rN)$. Our experimental shows that on average the 2D-SRC algorithm is $2 \sim 3$ times faster than 1D-SRC, more details can be seen in our experimental results.

As for the incremental SVD based version, its complexity also consists of two parts: let $n = size(Q_n)$, then the binary searching steps for eigenvalues are at most $C \times n$ loops, where C is the maximum possible searching steps, and each loop contains $O(n)$ float-point operations (flops); and the eigenvector computing step takes up $O(n^2)$ flops on computing all eigenvectors and it also needs an extra $O(n^{2+\delta})$ step for matrix-to-matrix multiplication (herein $0 \le \delta \le 1$ has not been well determined in computational complexity theory, but a $\delta \approx 0.807\ldots$ algorithm has been presented in [47]; however, a naive implementation of multiplying two matrices will take n^3 operations for $\delta = 1$). Thus, totally it takes $O(n^{2+\delta})$ flops and can be more efficient than an independent SVD which generally needs $O(n^3)$ flops. We summarize the complexity comparisons in Table 1.

Table 1. Complexity of algorithms

	SVD flops	ℓ_1-norm problem size
1D-SRC	–	$O(LN)$
2D-SRC	$O(N^3)$	$O(rN)$
2D-SRC-incremental	$O(N^{2+\delta})^*$	$O(rN)$

* here $0 \le \delta \le 1$ is still not determined in theory.

3. **Experiments.**

To evaluate the efficiency of the proposed algorithms in this paper, we compare their computational costs and recognition rates with the 1D-SRC algorithm on several face recognition tasks. By using several benchmark face databases, we show that the proposed two algorithms (2D-SRC and 2D-SRC-incremental) are faster than the 1D-SRC algorithm, and recognition accuracy is maintained. Next, we present our experimental results in detail.

3.1. **Datasets and configurations.** We have performed experiments on three face databases: ORL [48], Yale B plus Extended [49, 50] and AR face database [51]. For each database, alignments (e.g., fixing positions for the two eyes, cropping) are performed.

For the ORL face database, it contains 40 individuals, each with 10 face images, so totally 400 face images are available; the original image size of the database is 92×112, and we have cropped it into the size of 90×90.

For the Yale B plus Extended face database, it contains 38 individuals, each with many variants of poses and illuminations, but in our experiments we only adopted these images with pose of frontal view (corresponding to those with filenames as '$^*\_P00^*$') and all illuminations in subset 1 and 2. We crop and resize them into 90×90.

For the AR face database, we choose 120 individuals, each with 26 images available. We also crop and resize them into 90×90.

An important advantage claimed for 1D-SRC model is its robust performance, so we design our experiments containing two parts: one part uses image datasets without corruptions both for training and testing, the other part uses corrupted image datasets for testing but not in training.

Although many ℓ_1-optimization solvers are publicly available (e.g., [52, 53, 54]), ℓ1-magic [52] is chosen in [32]. More than that, in our experiments we also choose another ℓ_1-optimization solver $l1\_ls$ to solve the ℓ_1-norm minimization problem. The purpose is to make broad comparisons and investigate the performance in diversity when the proposed algorithms are collaborating with different ℓ_1-optimization solvers.

All the results are obtained using MATLAB on a machine with Intel(R) Xeon(R) CPU 2.33GHz, 16GB of RAM, Windows Server 2003 Standard x64 Edition.

3.2. **1D-SRC vs. 2D-SRC accuracy and speed.** In this part, we compare two algorithms: the original 1D-SRC algorithm and the proposed 2D-SRC algorithm. We focus on two performance: the recognition accuracy R and the average computational time T, and they are defined as

$$R = \frac{n_c}{n}, \tag{46}$$

$$T = \frac{\sum_{i=1}^{n} t_i}{n} \tag{47}$$

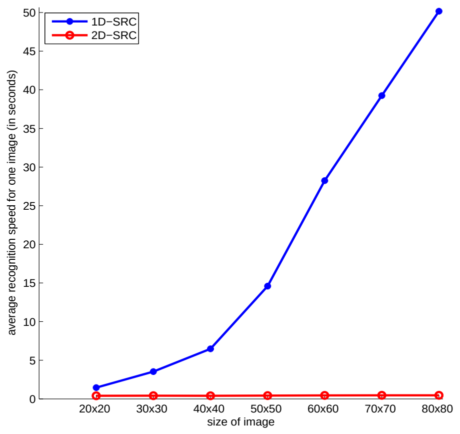
where n is the total number of testing samples, n_c is the number of correctly recognized testing samples, t_i is the computational time used for recognizing the ith test sample.

The experiments are performed as follows: given a database, we split it using a half-half training:testing ratio (i.e., 50% : 50%), and for each split, we choose 5 random repeats of the training/testing set and record the performance of these two algorithms, then average the 5 rounds performance as a final performance. To test the performance of 1D-SRC and 2D-SRC under different image sizes, we also repeat resizing the same split sets with varying sizes, and we have chosen the image size sequence as 20×20, 30×30, $\dots$, 80×80.

The experimental results are obtained on the ORL, the Yale B plus Extended (subset1+subset2), the AR face databases respectively. Figures 1-3 show the results when the databases are original and noise free. Figures 4-6 show the results when the testing samples are corrupted by adding small block noises randomly. The purpose of showing these two different type of results is to demonstrate both the 1D-SRC algorithm and the

image sizes	1D-SRC	2D-SRC
20×20	88.40	92.10
30×30	88.60	91.60
40×40	88.80	91.60
50×50	88.70	91.00
60×60	89.30	90.40
70×70	87.90	90.30
80×80	88.40	90.30

(A) Recognition rates

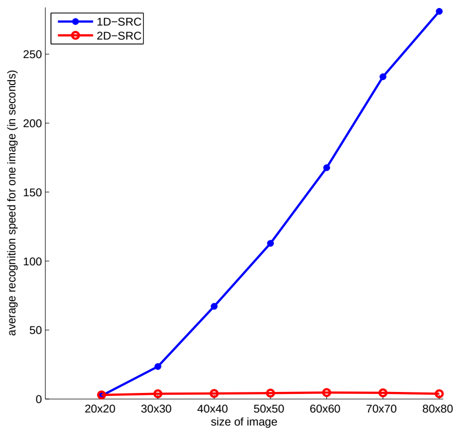


(B) Speed vs. dimensionality

FIGURE 1. Comparison of 1D-SRC and 2D-SRC on the ORL database

image sizes	1D-SRC	2D-SRC
20×20	100.00	100.00
30×30	100.00	100.00
40×40	100.00	100.00
50×50	100.00	100.00
60×60	100.00	100.00
70×70	100.00	100.00
80×80	100.00	100.00

(A) Recognition rates



(B) Speed vs. dimensionality

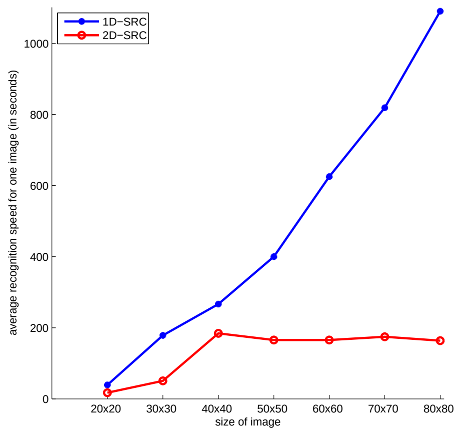
FIGURE 2. Comparison of 1D-SRC and 2D-SRC on the Yale B and Extended database

proposed 2D-SRC are robust when the testing data are corrupted by noises. In each figure, we show different recognition rates in a table, and draw the curve of computational time for different image dimensionality in a figure.

From these figures and tables, we conclude that: (1) The 2D-SRC algorithm can get a recognition rates close to the original 1D-SRC algorithm, in both cases (without noise/with noise). (2) The computational cost of the 2D-SRC algorithm is generally lower than that of 1D-SRC algorithm. More precisely, for small scale databases like the ORL and the Yale B plus Extended (subset1+subset2), the computational time costs of the original 1D-SRC algorithm are increasing as a linear function of the data dimensionality L, which is consistent with the $O(LN)$ complexity we have analyzed. But the computational time for the 2D-SRC algorithm is nearly constant because its complexity is $O(rN)$ and r is small in these experiments. For large-scale database like AR, the computational time for the original 1D-SRC algorithm is increasing similarly, but the time for the 2D-SRC algorithm is firstly increasing and then becomes flat. (3) The recognition rates of the 2D-SRC algorithmon is slightly higher than that of the 1D-SRC algorithm on the ORL database, the same on the Yale B plus Extended database and lower on

image sizes	1D-SRC	2D-SRC
20×20	86.50	86.12
30×30	86.40	86.08
40×40	86.90	86.40
50×50	86.86	86.50
60×60	89.90	86.60
70×70	87.10	86.90
80×80	87.50	87.20

(A) Recognition rates

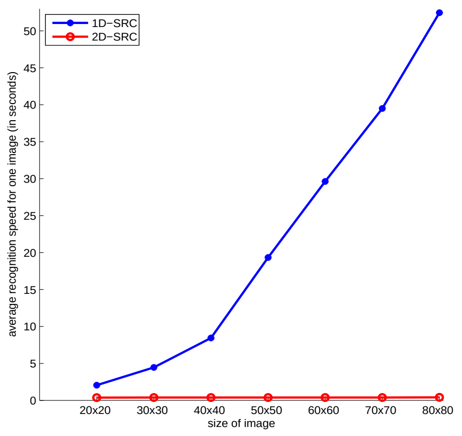


(B) Speed vs. dimensionality

FIGURE 3. Comparison of 1D-SRC and 2D-SRC on the AR database

image sizes	1D-SRC	2D-SRC
20×20	91.00	93.00
30×30	91.00	91.50
40×40	84.50	93.50
50×50	89.00	92.50
60×60	87.00	91.50
70×70	85.50	90.50
80×80	87.00	91.50

(A) Recognition rates



(B) Speed vs. dimensionality

FIGURE 4. Comparison of 1D-SRC and 2D-SRC on the ORL database (with corruptions)

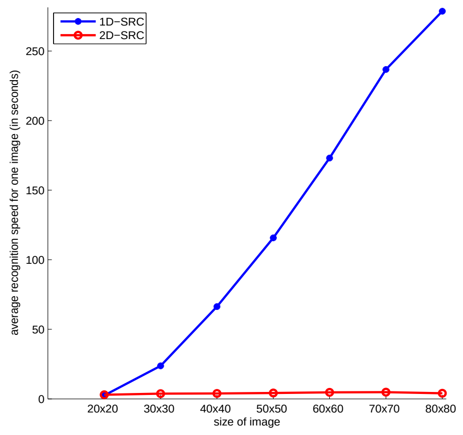
the AR database. Overall the recognition rates of the two algorithms are very close. Therefore, the 2D-SRC algorithm has competitive advantage in terms of of reducing the computational costs whilst maintaining a similar recognition performance.

3.3. **Performance and efficiency for incremental 2D-SRC.** In this part, we evaluate the performance of three algorithms (i.e., 1D-SRC, 2D-SRC, 2D-SRC-incremental) in the context of incremental learning. Also, the average recognition accuracy and computational time are adopted for evaluation.

The experiment is set up as follows. Given a database, we firstly divide it into a training set and a testing set using a ratio of $training : testing = 70\% : 30\%$; here we do not use the whole training set at once, but start from using a small set of them (e.g., 30% or so), and add the other training samples one by one to simulate a process with increasing number of training images; the testing set is kept unchanged all the time. For each intermediate training set, we apply the 1D-SRC, the 2D-SRC and the 2D-SRC-incremental algorithms respectively to do recognition and record their accuracy R and the average recognition time T for each sample.

image sizes	1D-SRC	2D-SRC
20×20	100.00	100.00
30×30	100.00	100.00
40×40	100.00	100.00
50×50	100.00	100.00
60×60	100.00	100.00
70×70	100.00	100.00
80×80	100.00	100.00

(A) Recognition rates

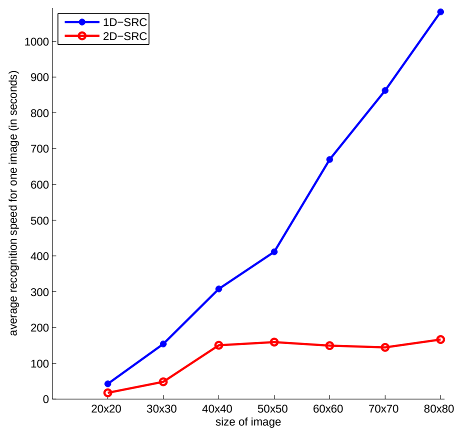


(B) Speed vs. dimensionality

FIGURE 5. Comparison of 1D-SRC and 2D-SRC on the Yale B and Extended database (with corruption)

image sizes	1D-SRC	2D-SRC
20×20	75.26	74.17
30×30	75.17	74.73
40×40	75.65	75.03
50×50	76.10	75.13
60×60	76.20	75.70
70×70	76.45	75.90
80×80	76.51	76.10

(A) Recognition rates



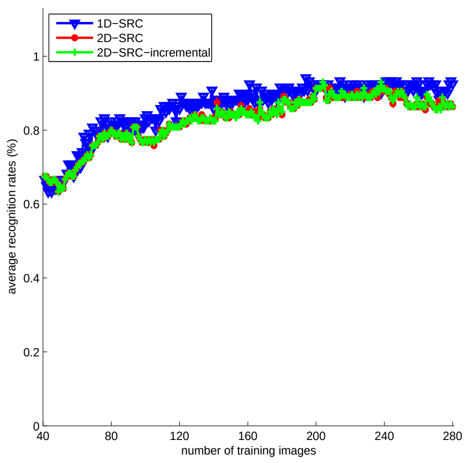
(B) Speed vs. dimensionality

FIGURE 6. Comparison of 1D-SRC and 2D-SRC on the AR database (with corruptions)

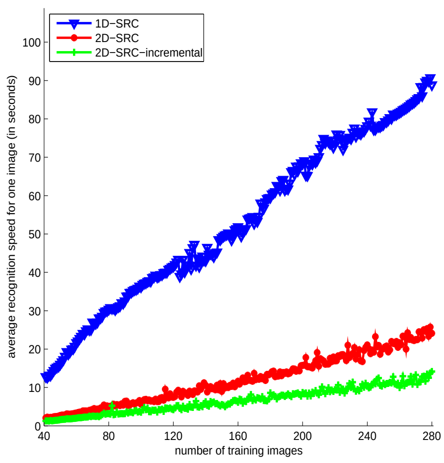
The experimental results on the ORL database, the Yale B plus Extended (subset1+subset2) database and the AR database are shown in Figures 7-9 respectively. In each figure, on the left we show the recognition accuracy difference of the three algorithms, when the number of training samples increases, and on the right we show their corresponding average computational time trends as the number of training samples varies.

From the figures, we can see consistently that for all three databases, the 1D-SRC algorithm always uses the most computational time, the 2D-SRC algorithm use less time than the 1D-SRC algorithm, and the 2D-SRC-incremental algorithm use the least time among three algorithms. The time consuming gap between the 1D-SRC algorithm and the 2D-SRC algorithm is much bigger that the gap between the 2D-SRC algorithm and the 2D-SRC-incremental algorithm.

Also, the recognition rates of the three algorithms are very close, mostly the 1D-SRC algorithm is slightly higher than the 2D-SRC algorithm and the 2D-SRC-incremental algorithm, but sometimes the contrary is the case. It can be seen that there is nearly no difference of recognition rates between the 2D-SRC algorithm and the 2D-SRC-incremental algorithm. This can be expected since the incremental SVD procedure employed in the

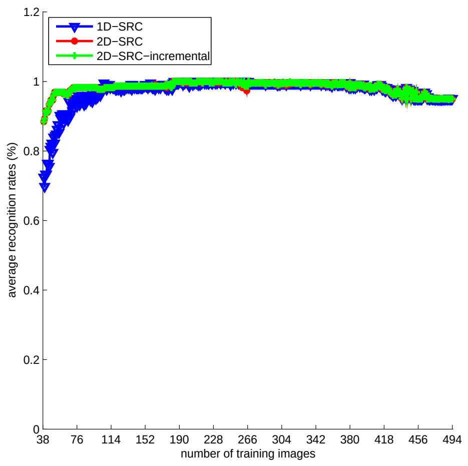


(A) Recognition rates vs. number of training samples

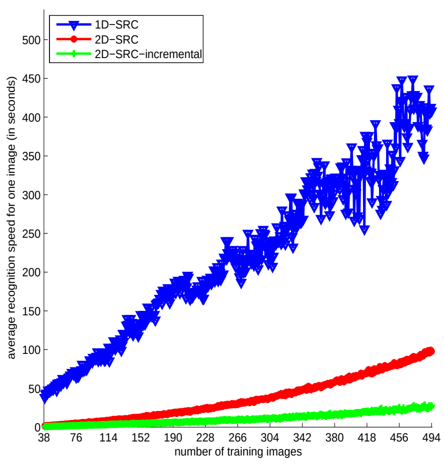


(B) Computational time vs. number of training samples

FIGURE 7. Incremental learning results on the ORL face database



(A) Recognition rates vs. number of training samples



(B) Computational time vs. number of training samples

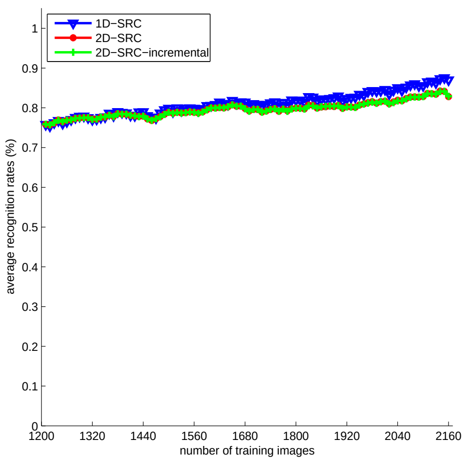
FIGURE 8. Incremental learning results on the Yale B and Extended face database

2D-SRC-incremental algorithm is a pure computational acceleration with careful design for not losing accuracy.

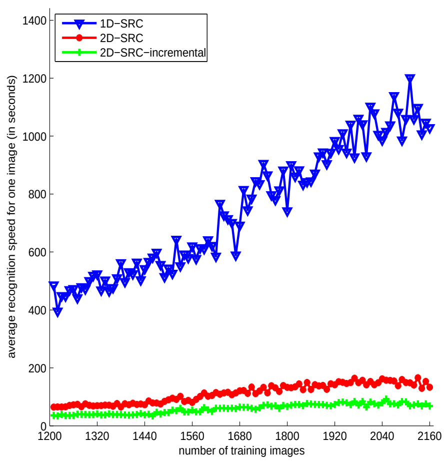
As a result, we conclude that applying incremental procedure on 2D-SRC is helpful in reducing the computational time further, but not as much as the cost reduction from 1D-SRC to 2D-SRC, and the performance for the three algorithms are similar.

3.4. **Comparisons of two ℓ_1-solvers.** In this section, we use two different ℓ_1-solvers for comparison. In fact, the performance of the proposed algorithms is relying on the performance of the ℓ_1-solver used. In order to demonstrate the effectiveness of the proposed algorithms in using different ℓ_1-solvers, we choose two ℓ1-solvers (i.e., ℓ1-magic and *l*1_*ls*), then execute 1D-SRC and 2D-SRC separately to record their performance. The experiment setup is the same as in previous Section 3.2.

The results on three databases, i.e., the ORL database, the Yale B plus Extended (subset1+subset2) database and the AR database, are shown in Figures 10-12, when all



(A) Recognition rates vs. number of training samples

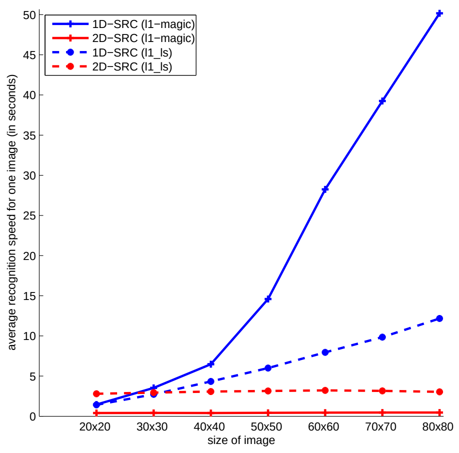


(B) Computational time vs. number of training samples

FIGURE 9. Incremental learning results on the AR face database

image sizes	ℓ1-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20 × 20	88.40	92.10	90.40	91.80
30 × 30	88.60	91.60	89.10	91.80
40 × 40	88.80	91.60	89.60	91.70
50 × 50	88.70	91.00	89.50	90.80
60 × 60	89.30	90.40	89.30	90.70
70 × 70	87.90	90.30	89.00	90.40
80 × 80	88.40	90.30	89.10	90.50

(A) Recognition rates



(B) Speeds vs. dimensionality

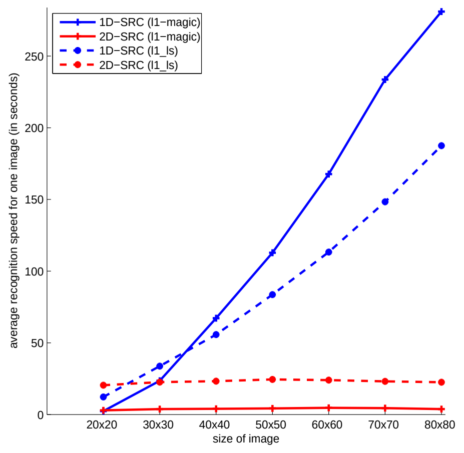
FIGURE 10. Comparison of different ℓ_1-solvers on the ORL database

the training and testing images data are noise free. In Figures 13-15, the results are shown when the testing images data are corrupted with noise and the training images data remain the same. In each figure, we show on the left the recognition accuracy of four combinations: two of them are from the 1D-SRC algorithm using ℓ1-magic and $l1_ls$ as solvers respectively, the other two are from the 2D-SRC algorithm also using ℓ1-magic and $l1_ls$ as solvers respectively. Different image sizes of the same database have been extensively tested. On the right, we show their average computational time trend when the image size varies.

From the experimental results, we conclude that: (1) the performance (recognition rate and computation time) is different between two different solvers, and the $l1\_ls$ solver is generally performs better than the ℓ1-magic solver. (2) For small scale databases, like the ORL and the Yale B plus Extended (subset1+subset2), the 2D-SRC algorithm has consistent computational improvement for both ℓ_1-solvers, which demonstrates the effectiveness of the 2D-SRC algorithm. (3) For large scale databases, like the AR, for both ℓ_1-solvers, the computational time of the 2D-SRC algorithm is firstly increasing and then

image sizes	$\ell 1$-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20×20	100.00	100.00	100.00	100.00
30×30	100.00	100.00	100.00	100.00
40×40	100.00	100.00	100.00	100.00
50×50	100.00	100.00	100.00	100.00
60×60	100.00	100.00	100.00	100.00
70×70	100.00	100.00	100.00	100.00
80×80	100.00	100.00	100.00	100.00

(A) Recognition rates

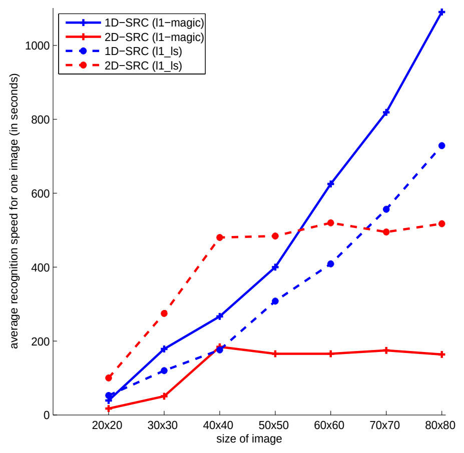


(B) Speeds vs. dimensionality

FIGURE 11. Comparison of different ℓ_1-solvers on the Yale B plus Extended database

image sizes	$\ell 1$-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20×20	86.50	86.12	90.00	89.90
30×30	86.40	86.08	90.37	90.13
40×40	86.90	86.40	90.50	89.96
50×50	86.86	86.50	90.38	90.10
60×60	89.90	86.60	90.85	90.25
70×70	87.10	86.90	90.90	90.55
80×80	87.50	87.20	91.28	91.06

(A) Recognition rates



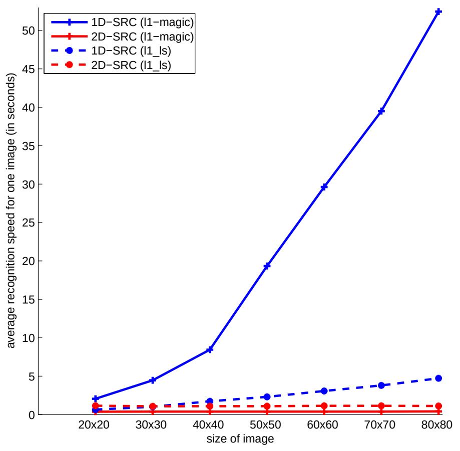
(B) Speeds vs. dimensionality

FIGURE 12. Comparison of different ℓ_1-solvers on the AR database

becomes flat, but the improvement percentages are different between these two ℓ_1-solvers. It can be seen that for $\ell 1$-magic, the 2D-SRC algorithm always computes faster than the 1D-SRC algorithm; however, for the $l1\_ls$, the 2D-SRC algorithm firstly requires more computational time for small problems; however, the 2D-SRC algorithm becomes faster than the 1D-SRC algorithm as the size of problems grows larger. From this analysis, one can see that the $l1\_ls$ solver is more optimized for computation. (4) Regarding the recognition rates, for both the 1D-SRC algorithm and the 2D-SRC algorithm, the experimental results show that the $l1\_ls$ solver is always having higher recognition rates than the $\ell 1$-magic solver on the same database, which indicates that the $l1\_ls$ solver could be a better choice for these recognition tasks. While the $\ell 1$-solver is fixed (either $\ell 1$-magic or $l1\_ls$), on the ORL database the 2D-SRC algorithm has slightly higher recognition rates than the 1D-SRC algorithm, on the Yale B plus Extended database they have the same recognition rates, and on the AR database the 1D-SRC algorithm has negligible higher recognition rates than the 2D-SRC algorithm. Overall the recognition rates of the two algorithms are close provided they are using the same $\ell 1$-solver. In conclusion, the 2D-SRC algorithm demonstrates its effectiveness for these two ℓ_1-solvers.

image sizes	$\ell 1$-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20×20	91.00	93.00	92.50	93.00
30×30	91.00	91.50	90.50	91.50
40×40	84.50	93.50	90.00	93.00
50×50	89.00	92.50	89.00	92.50
60×60	87.00	91.50	88.50	91.50
70×70	85.50	90.50	89.00	91.00
80×80	87.00	91.50	89.00	92.00

(A) Recognition rates

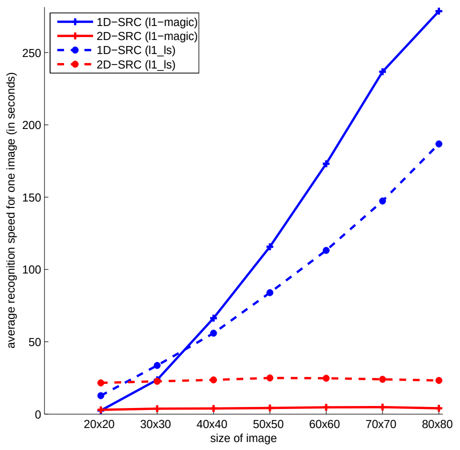


(B) Speeds vs. dimensionality

FIGURE 13. Comparison of different ℓ_1-solvers on the ORL database (with corruptions)

image sizes	$\ell 1$-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20×20	100.00	100.00	100.00	100.00
30×30	100.00	100.00	100.00	100.00
40×40	100.00	100.00	100.00	100.00
50×50	100.00	100.00	100.00	100.00
60×60	100.00	100.00	100.00	100.00
70×70	100.00	100.00	100.00	100.00
80×80	100.00	100.00	100.00	100.00

(A) Recognition rates

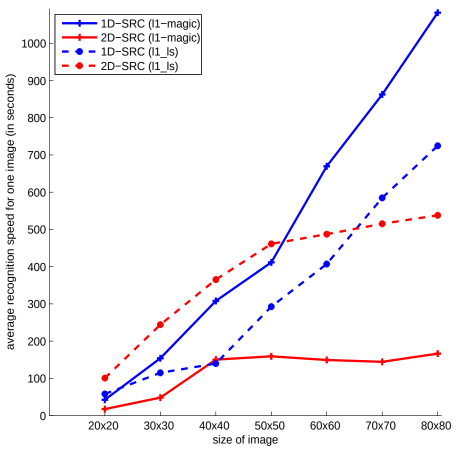


(B) Speeds vs. dimensionality

FIGURE 14. Comparison of different ℓ_1-solvers on the Yale B+Extended (with corruptions)

image sizes	$\ell 1$-magic		$l1\_ls$	
	1D-SRC	2D-SRC	1D-SRC	2D-SRC
20×20	75.26	74.17	81.95	81.05
30×30	75.17	74.73	81.96	81.10
40×40	75.65	75.03	82.05	81.37
50×50	76.10	75.13	82.26	81.46
60×60	76.20	75.70	82.65	81.63
70×70	76.45	75.90	82.90	81.80
80×80	76.51	76.10	83.18	82.10

(A) Recognition rates



(B) Speeds vs. dimensionality

FIGURE 15. Comparison of different ℓ_1-solvers on the AR database (with corruptions)

4. **Conclusion.** In this paper, we have proposed two fast sparse representation algorithms for robust face recognition. One is the 2D-SRC algorithm and the other is an incremental version of 2D-SRC. The main idea for these these two algorithms is based on the inner product matrix computation, and such computation can be performed in an incremental manner. Experimental results on different benchmark datasets as well as different ℓ_1-optimization solvers show that the proposed methods are significantly faster than the original SRC algorithm whilst still maintain or even enhance the recognition rate, especially for large datasets or high-resolution images.

This work is an extension of our previous conference paper in ICPR 2010 [55].

Acknowledgement. The authors thank the anonymous referees for their valuable suggestions and helpful comments which help improve the quality of the paper. This project was partly supported by the NSFC with grant number U0835005, 60803083, the Guangdong Program with Grant No. 2009A090100025 and Australia Research Council.

REFERENCES

[1] W. Zhao, R. Chellappa, P. J. Phillips and A. Rosenfeld, Face recognition: A literature survey, *ACM Computing Surveys*, vol.35, no.4, pp.399-458, 2003.

[2] J. P. Phillips, H. Moon, S. A. Rizvi and P. J. Rauss, The FERET evaluation methodology for face-recognition algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, no.10, pp.1090-1104, 2000.

[3] A. J. O'Toole, P. J. Phillips and A. Narvekar, FRVT 2002 evaluation report, *Technical Report 6965*, NISTIR, 2003.

[4] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott and M. Sharpe, FRVT 2006 and ice 2006 large-scale experimental results, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.32, no.5, pp.831-846, 2010.

[5] X. Wang and X. Tang, A unified framework for subspace face recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.26, no.9, pp.1222-1228, 2004.

[6] C.-Y. Chang and H.-R. Hsu, Application of principal component analysis to a radial-basis function committee machine for face recognition, *International Journal of Innovative Computing, Information and Control*, vol.5, no.11(B), pp.4145-4154, 2009.

[7] S. D. Lin, J.-H. Lin and C.-C. Chiang, Combining scale invariant feature transform with principal component analysis in face recognition, *ICIC Express Letters*, vol.3, no.4(A), pp.927-932, 2009.

[8] A. V. Nefian and M. H. Hayess, Hidden markov models for face recognition, *The International Conference on Acoustics, Speech and Signal Processing*, Seattle, Washington, pp.2721-2724, 1998.

[9] B. Moghaddam, Bayesian face recognition, *Pattern Recognition*, vol.33, no.11, pp.1771-1782, 2000.

[10] B. Heisele, P. Ho and T. Poggio, Face recognition with support vector machines: Global versus component-based approach, *The 8th IEEE International Conference on Computer Vision*, vol.2, pp.688-694, 2001.

[11] M.-H. Yang, Face recognition using kernel methods, *Advances in Neural Information Processing Systems*, vol.14, pp.1457-1464, 2001.

[12] M.-H. Yang, Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods, *The 5th IEEE International Conference on Automatic Face and Gesture Recognition*, Washington, pp.215-220, 2002.

[13] J. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, Face recognition using kernel direct discriminant analysis algorithms, *IEEE Transactions on Neural Networks*, vol.14, no.1, pp.117-126, 2003.

[14] J.-B. Li, Nonparametric kernel discriminant analysis for face recognition under varying lighting conditions, *ICIC Express Letters*, vol.4, no.3(B), pp.999-1004, 2010.

[15] I G. P. S. Wijaya, K. Uchimura and Z. Hu, Face recognition based on dominant frequency features and multiresolution metric, *International Journal of Innovative Computing, Information and Control*, vol.5, no.3, pp.641-652, 2009.

[16] L. Wiskott, J. M. Fellous, N. Kuiger and C. von der Malsburg, Face recognition by elastic bunch graph matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19, no.7, pp.775-779, 1997.

[17] T. F. Cootes, K. Walker and C. J. Taylor, View-based active appearance models, *The 4th IEEE International Conference on Automatic Face and Gesture Recognition*, pp.227-232, 2000.

[18] V. Blanz and T. Vetter, Face recognition based on fitting a 3d morphable model, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.25, no.9, pp.1063-1074, 2003.

[19] M. Turk and A. Pentland, Eigenfaces for recognition, *Journal of Cognitive Neuroscience*, vol.3, no.1, pp.71-86, 1991.

[20] A. M. Martinez and A. C. Kak, PCA versus LDA, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.23, no.2, pp.228-233, 2001.

[21] P. N. Belhumeur, J. Hespanha and D. J. Kriegman, Eigenfaces vs. fisherfaces: Recognition using class specific linear projection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19, no.7, pp.711-720, 1997.

[22] J. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, Face recognition using lda-based algorithms, *IEEE Transactions on Neural Networks*, vol.14, no.1, pp.195-200, 2003.

[23] X. He, S. Yan, Y. Hu, P. Niyogi and H. J. Zhang, Face recognition using Laplacianfaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.27, no.3, pp.328-340, 2005.

[24] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang and S. Lin, Graph embedding and extensions: A general framework for dimensionality reduction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.29, no.1, pp.40-51, 2007.

[25] M. S. Bartlett, J. R. Movellan and T. J. Sejnowski, Face recognition by independent component analysis, *IEEE Transactions on Neural Networks*, vol.13, no.6, pp.1450-1464, 2002.

[26] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, *Nature*, vol.401, no.6755, pp.788-791, 1999.

[27] S. Z. Li, X. W. Hou, H. J. Zhang and Q. S. Cheng, Learning spatially localized, parts-based representation, *The IEEE Conference on Computer Vision and Pattern Recognition*, vol.1, pp.I-207-I-212, 2001.

[28] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min and W. Worek, Overview of the face recognition grand challenge, *The IEEE Conference on Computer Vision and Pattern Recognition*, vol.1, pp.947-954, 2005.

[29] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer and W. Worek, Preliminary face recognition grand challenge results, *The 7th International Conference on Automatic Face and Gesture Recognition*, pp.15-24, 2006.

[30] P. J. Phillips, W. T. Scruggs, A. J. O'toole, P. J. Flynn, W. Kevin, C. L. Schott and M. Sharpe, FRVT 2006 and ICE 2006 large-scale results, *Technical Report*, 2007.

[31] R. Chellappa, P. Sinha and P. J. Phillips, Face recognition by computers and humans, *Computing Now*, 2010.

[32] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma, Robust face recognition via sparse representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.31, no.2, pp.210-227, 2009.

[33] E. J. Candes, J. Romberg and T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. on Information Theory*, vol.52, no.2, pp.489-509, 2006.

[34] D. L. Donoho, Compressed sensing, *IEEE Trans. on Information Theory*, vol.52, no.4, pp.1289-1306, 2006.

[35] H. Zou, T. Hastie and R. Tibshirani, Sparse principal component analysis, *Journal of Computational and Graphical Statistics*, vol.15, no.2, 2006.

[36] J. Mairal, M. Elad and G. Sapiro, Sparse representation for color image restoration, *IEEE Transactions on Image Processing*, vol.17, no.1, pp.53-69, 2008.

[37] A. Pagnani, F. Tria and M. Weigt, Classification and sparse-signature extraction from gene-expression data, *ArXiv Preprint*, 2009.

[38] Rice University, *Compressive Sensing Resources*, http://dsp.rice.edu/cs.

[39] J. J. Fuchs, On sparse representations in arbitrary redundant bases, *IEEE Transactions on Information Theory*, vol.50, no.6, pp.1341-1344, 2004.

[40] J.-J. Fuchs, Fast implementation of a ℓ_1-ℓ_1 regularized sparse representations algorithm, *The IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp.3329-3332, 2009.

[41] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[42] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990.

[43] D. P. O'Leary and G. W. Stewart, Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices, *Journal of Computational Physics*, vol.90, no.2, pp.497-505, 1990.

[44] K. Dickson and T. Selee, Eigenvectors of arrowhead matrices via the adjugate, *Preprint*, 2007.
[45] F. Diele, N. Mastronardi, M. van Barel and E. van Camp, On computing the spectral decomposition of symmetric arrowhead matrices, *Computational Science and Its Applications, LNCS*, vol.3044, pp.932-941, 2004.
[46] L. Shen and B. W. Suter, Bounds for eigenvalues of arrowhead matrices and their applications to hub matrices and wireless communications, *EURASIP Journal on Advances in Signal Processing*, 2009.
[47] V. Strassen, Gaussian elimination is not optimal, *Numer. Math.*, vol.13, pp.354-356, 1969.
[48] *The ORL Database of Faces*, AT&T Laboratories Cambridge.
[49] A. S. Georghiades, D. J. Kriegman and P. N. Belhurneur, Illumination cones for recognition under variable lighting: Faces, *The IEEE Conference on Computer Vision and Pattern Recognition*, pp.52-58, 1998.
[50] K.-C. Lee, J. Ho and D. J. Kriegman, Acquiring linear subspaces for face recognition under variable lighting, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.27, no.5, pp.684-698, 2005.
[51] A. M. Martinez and R. Benavente, The AR face database, *CVC Tech. Report #24*, 1998.
[52] E. Candes and J. Romberg, *l1-magic Code*, http://www.acm.caltech.edu/l1magic/.
[53] S.-J. K. K. Koh and S. Boyd, *l1_ls Code*, http://www.stanford.edu/ boyd/l1_ls/.
[54] S. B. J. Bobin and E. Cands, *NESTA Code*, http://www.acm.caltech.edu/ nesta/.
[55] H. Qiu, D.-S. Pham, S. Venkatesh, W. Liu and J.-H. Lai, A fast extension for sparse representation on robust face recognition, *The 20th International Conference on Pattern Recognition*, pp.1023-1027, 2010.

Appendix.

Lemma 1. *Let Q be the Frobenius inner product in (14). $rank(Q) \le \min(L, N)$, where $L = hw$ is the total pixel number of an $h \times w$ image and N is the number of training images.*

Proof: $rank(Q) \le N$ since

$$Q = \begin{bmatrix} \langle A_1, A_1 \rangle_F & \cdots & \langle A_1, A_N \rangle_F \\ \vdots & \ddots & \vdots \\ \langle A_N, A_1 \rangle_F & \cdots & \langle A_N, A_N \rangle_F \end{bmatrix}_{N\times N} \tag{48}$$

$rank(Q) \le L$ since

$$Q = \begin{bmatrix} vec\left(A_1\right)^T vec\left(A_1\right) & \cdots & vec\left(A_1\right)^T vec\left(A_N\right) \\ \vdots & \ddots & \vdots \\ vec\left(A_N\right)^T vec\left(A_1\right) & \cdots & vec\left(A_N\right)^T vec\left(A_N\right) \end{bmatrix}_{N\times N} \tag{49}$$

$$= \begin{bmatrix} vec\left(A_1\right)^T \\ \vdots \\ vec\left(A_N\right)^T \end{bmatrix}_{N\times L} \begin{bmatrix} vec\left(A_1\right) & \cdots & vec\left(A_N\right) \end{bmatrix}_{L\times N} \tag{50}$$

where $vec\left(\cdot\right)$ is the vectorizing operator of matrix.

Lemma 2. *If $\mathbf{Q}$ is a positive semi-definite matrix, then all its eigenvalues are non-negative.*

Proof: Suppose λ is an arbitrary eigenvalue of $\mathbf{Q}$, associated with an eigenvector $\mathbf{v} \neq \mathbf{0}$. Thus by definition $\mathbf{Qv} = \lambda\mathbf{v}$, which implies $\mathbf{v}^T\mathbf{Qv} = \lambda\mathbf{v}^T\mathbf{v}$. Since $\mathbf{Q}$ is positive semi-definite, we have

$$\lambda = \frac{\mathbf{v}^T\mathbf{Qv}}{\mathbf{v}^T\mathbf{v}} \ge 0.$$

Lemma 3. *In Theorem 2.1, it is assumed that all eigenvalues $\{\lambda_i\}_{i=1}^n$ in (39) are distinct from each other and all $z_i \neq 0$. However, when some eigenvalues λ_i are repeated and some $z_i = 0$, computation can be further simplified and investigated rigorously. We have the following complementary results:*

(1) If $z_i = 0$ in R_{n+1}, then λ_i must be an eigenvalue of R_{n+1}. By deleting the i^{th} row and i^{th} column from R_{n+1} we can get a smaller matrix.

$$\tilde{\mathbf{R}}_{(n-1)+1} = \begin{pmatrix} \lambda_1 & & & & & & z_1 \\ & \ddots & & & & & \vdots \\ & & \lambda_{i-1} & & & & z_{i-1} \\ & & & \lambda_{i+1} & & & z_{i+1} \\ & & & & \ddots & & \vdots \\ & & & & & \lambda_n & z_n \\ z_1 & \cdots & z_{i-1} & z_{i+1} & \cdots & z_n & c \end{pmatrix} \tag{51}$$

suppose the EVD of $\tilde{\mathbf{R}}_{(n-1)+1}$ is given by $\tilde{\mathbf{R}}_{(n-1)+1} = \tilde{\mathbf{V}}_{(n-1)+1}\tilde{\mathbf{D}}_{(n-1)+1}\tilde{\mathbf{V}}_{(n-1)+1}^T$, then the EVD of R_{n+1} is

$$\tilde{\mathbf{D}}_{n+1} = \begin{pmatrix} \lambda_i & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{D}}_{(n-1)+1} \end{pmatrix} \tag{52}$$

$$\tilde{\mathbf{V}}_{n+1} = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{V}}_{(n-1)+1} \end{pmatrix} \tag{53}$$

$$\mathbf{R}_{n+1} = \tilde{\mathbf{V}}_{n+1}\tilde{\mathbf{D}}_{n+1}\tilde{\mathbf{V}}_{n+1}^T \tag{54}$$

notice that $\tilde{\mathbf{R}}_{(n-1)+1}$ still has the "arrowhead" form like $\mathbf{R}_{n+1}$, so the above argumentation can be used on $\tilde{\mathbf{R}}_{(n-1)+1}$ recursively until there is no $z_i = 0$ in the matrix. Finally, we can obtain a $(p+1)\times(p+1)$ "arrowhead" matrix $\tilde{\mathbf{R}}_{p+1}$ just like $\mathbf{R}_{n+1}$ but with all $z_i \neq 0$, and we go on finding its EVD in Step (2).

(2) After Step (1), we can now assume for $\tilde{\mathbf{R}}_{p+1}$ all its $z_i \neq 0$. If $\tilde{\mathbf{R}}_{p+1}$ has no repeated λ_k values, then the condition of Theorem 2.1 is satisfied, and its EVD can be computed exactly as Theorem 2.1 shows. Otherwise, go further to Step (3).

(3) If $\tilde{\mathbf{R}}_{p+1}$ has some repeated λ_k values, let $\lambda_k = \lambda_{k+1} = \cdots = \lambda_{k+q} = \hat{\lambda}$ $(q \geq 1)$ is an eigen value that repeats $q+1$ times, then $\hat{\lambda}$ must be an eigenvalue of $\mathbf{R}_{n+1}$ with multiplicity q, and associated with q mutually orthnormal eigenvectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_q}$ which can be found as:

$$v_i^{(j)} = \begin{cases} \dfrac{z_{k+i}z_{k+j}}{z_k^2 + z_{k+1}^2 + \cdots + z_{k+j-1}^2}, & i = 0, \ldots, j-1 \\ -1, & i = j \\ 0, & \textit{otherwise} \end{cases} \tag{55}$$

where $v_i^{(j)}$ is the j^{th} entry of the eigenvector $\mathbf{v_i}$. All duplicate values in $\{\lambda_k\}_{k=1}^p$ can be removed to make a unique sequence $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_r$, with $q_1+1, q_2+2, \ldots, q_r+1$ ($q_l \geq 0$ for $\forall l$) being their multiplicity, then the above procedure can be applied to each $\hat{\lambda}$ (if $q_k = 0$ then it can be skipped) and totally $q_1 + q_2 + \cdots + q_r \equiv p - r$ eigenvalue-eigenvector pairs of $\tilde{\mathbf{R}}_{p+1}$ can be found, the other $r+1$ eigenvalues are exactly the roots of following equation with respect to d.

$$\frac{\hat{z}_1^2}{d - \hat{\lambda}_1} + \cdots + \frac{\hat{z}_r^2}{d - \hat{\lambda}_r} = d - c \tag{56}$$

where $\hat{z}_k^2 = \sum\limits_{if \lambda_i == \hat{\lambda}_k} z_i^2$ *is the sum of all appearing* z_i^2 *corresponding to the repeated value* $\hat{\lambda}_k$*. This equation can be solved just like (39) in Theorem 2.1 since it has* $r+1$ *distinct roots* $d_1, d_2, \ldots, d_r$*. The eigenvector related to* d_k *is the same one given by (41) in Theorem 2.1. So, all* $(p-r)+(r+1) = p+1$ *eigenvalues and eigenvectors of* $\tilde{\mathbf{R}}_{p+1}$ *can be computed now, even if it has repeated eigenvalues.*

(4) All the computations in (1)-(3) are explicitly done by explicit formulas, and their complexity is $O(n)$ *for computing one eigenvalue and one eigenvector. So, the complexity of finding all eigenvalue and eigenvectors will be* $O(n^2)$*.*

Proof: We refer the reader for proving (1)-(3) in [45], and (4) is obvious by checking each computation.