Is face recognition really a Compressive Sensing problem?

Qinfeng Shi†, Anders Eriksson†, Anton van den Hengel†, Chunhua Shen† ‡
The Australian Centre for Visual Technologies, The university of Adelaide †
National ICT Australia ‡
{javen.shi, anders.eriksson, anton.vandenhengel, chunhua.shen} @adelaide.edu.au

Abstract

Compressive Sensing has become one of the standard methods of face recognition within the literature. We show, however, that the sparsity assumption which underpins much of this work is not supported by the data. This lack of sparsity in the data means that compressive sensing approach cannot be guaranteed to recover the exact signal, and therefore that sparse approximations may not deliver the robustness or performance desired. In this vein we show that a simple ℓ_2 approach to the face recognition problem is not only significantly more accurate than the state-of-the-art approach, it is also more robust, and much faster. These results are demonstrated on the publicly available YaleB and AR face datasets but have implications for the application of Compressive Sensing more broadly.

1. Introduction

The application of Compressive Sensing (CS) to the problem of face recognition has received significant recent attention in the literature (see [18, 19, 14] for example). The goal of many such methods has been to exploit the underlying sparsity in the problem in order to improve the robustness, speed, or accuracy with which classification might be performed, or all three. As in many applications of CS, however, the sparsity of in the problem is assumed, rather than proven, or measured. We show here that the sparsity assumption is not supported by data, and that an ℓ_2-based approach out-performs the state-of-the-art in CS methods in terms of speed, accuracy, and robustness. The implications are important for the application of CS to face recognition but also to other problems where sparsity is assumed rather than proven.

2. The space of all face images

Consider face recognition with n frontal training face images collected from K subjects. Let n_k denote the number of training images of subject k , thus $n = \sum_{k=1}^{K} n_k$. Without loss of generality, we assume that all the data have been sorted according to their labels and then we collect all the vectors in a single matrix $\mathbf{A}$ with m rows and n columns, given by

$$\mathbf{A} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{m \times n}. \tag{1}$$

The assumption which underpins the application of CS to face recognition by Yang *et al.* [19], Wright *et al.* [18], and Shi *et al.* [14] is as follows:

Assumption 1 *Any test image*[1] *lies in the subspace spanned by the training images belonging to the same person. That is for any test image* $\mathbf{x}$*, without knowing its label information, it's assumed that there exists a* η*-sparse*[2] $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ *such that*

$$\mathbf{x} = \mathbf{A}\boldsymbol{\alpha}. \tag{2}$$

To seek a sparse solution, one could use

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\boldsymbol{\alpha}\|_{\ell_1} \tag{3a}$$

$$\text{s.t. } \mathbf{x} = \mathbf{A}\boldsymbol{\alpha}. \tag{3b}$$

Solving this problem via linear programming becomes computationally expensive when m is large, however.

In order to exploit the presumed sparsity in the problem the authors in [19] and [18] generate a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{d \times m}$ (where $d \ll m$) and identify the vector $\boldsymbol{\alpha}$ which minimises the following ℓ_1 problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\boldsymbol{\alpha}\|_{\ell_1} \tag{4a}$$

$$\text{s.t. } \boldsymbol{\Phi}\mathbf{x} = \boldsymbol{\Phi}\mathbf{A}\boldsymbol{\alpha}, \tag{4b}$$

or the related problem:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\boldsymbol{\alpha}\|_{\ell_1} \tag{5a}$$

$$\text{s.t. } \|\boldsymbol{\Phi}\mathbf{x} - \boldsymbol{\Phi}\mathbf{A}\boldsymbol{\alpha}\|_{\ell_2} \leq \varepsilon, \tag{5b}$$

[1] The image here can be either an original image, or a feature image extracted from the original one, *e.g.* an eigenface. Often the dimensionality of the feature has to be reduced due to the complexity of the recognition algorithm.

[2] A n-dimensional signal is said η-sparse if it has at most η non-zero entries.

for a given error tolerance $\varepsilon > 0$. Introducing the matrix $\mathbf{\Phi}$ significantly reduces the computational complexity (particularly when $d \ll m$), yet the CS signal recovery theorem [5, 4, 11] shows that when $d \geq O(\eta \log(n/\eta))$ the signal $\boldsymbol{\alpha}$ can be exactly recovered (that is, it reaches the optimum of the original problem specified in Equation (3)) with overwhelming probability at least $1 - e^{O(-d)}$.

Shi *et al.* in [14] showed the connection between Hash Kernels [12, 13, 17] and CS. In doing so they showed that it is possible to replace $\mathbf{\Phi}$ with an *implicit* hash matrix $\mathbf{H}$ in order to reduce storage requirements and speed up face recognition with Orthogonal Matching Pursuit (OMP)[15].

2.1. Is the set of face images really sparse?

Despite the results of [19], [18] and [14], it is clear that the validity of Assumption 1 depends on the particular data set being used. What is not as immediately clear is that Assumption 1 does not hold for data of the form typically used to evaluate face recognition algorithms. Assumption 1 is sometimes justified on the basis of the result in [2] that the images of a rigid Lambertian surface under varying illumination lie close to a 9-dimensional linear subspace. This presumes perfect registration of the images, no self-shadowing, occlusion, or specularities, and ignores the fact that faces are neither rigid nor Lambertian, however. In order to evaluate the validity of Assumption 1 directly we form the matrix $\mathbf{A}$ in the same manner as in [19] and [18].

The AR dataset, which is used in many face recognition papers including those above, consists of 26 aligned images of each subject in different lighting conditions and with different facial expressions and disguises. We randomly selected 100 such subjects and cropped the images to 165×120 pixels and converted to grayscale (as in [19]), and, using (1), formed the matrix $\mathbf{A}$ where $m = 19800$ and $n = 2600$. A plot of the log of the singular values of this matrix is given in Figure 1.

Typically a subset (often half) of the database is used for training, and the remainder for testing. If Assumption 1 holds then we would hope that 13 training images per subject would suffice to span the space of all face images of the subject, and thus that the remaining (testing) 13 were linear combinations of the training set. This would lead to a matrix $\mathbf{A}$ with rank at most 1300 (rank 13 per subject for 100 subjects). Figure 1 does not support this hypothesis, however, as there is no obvious dip in the singular values of $\mathbf{A}$ at 1300, or any at other point. Note that Figure 1 depicts the singular values of the matrix of *all* face images in the AR data set, rather than only those for a single subject. It thus shows not only that there is little redundancy in AR dataset face images for any single subject, but also that there is little redundancy in AR dataset face images for *all* subjects collectively. The first few singular values are significantly higher than the remainder, which reflects the commonality in the overall shape of the face, but there is little differentiation between the remaining components.

The fact that there is no significant dip in the singular values of $\mathbf{A}$ does not discount sparsity completely, as there is inevitably noise in the training data, but it gives an indication that there is no simple linear dependence in the data set.

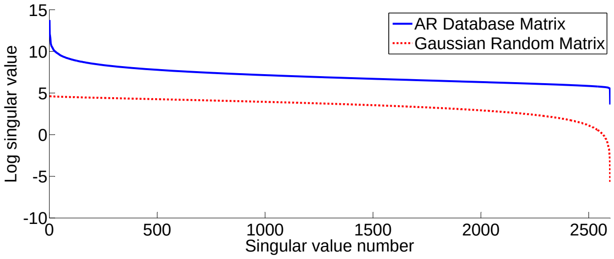


Figure 1. The log of the singular values of the data matrix $\mathbf{A}$ calculated using the AR data set, and for comparison, of a matrix of the same size with elements sampled from $\mathcal{N}(0, 1)$.

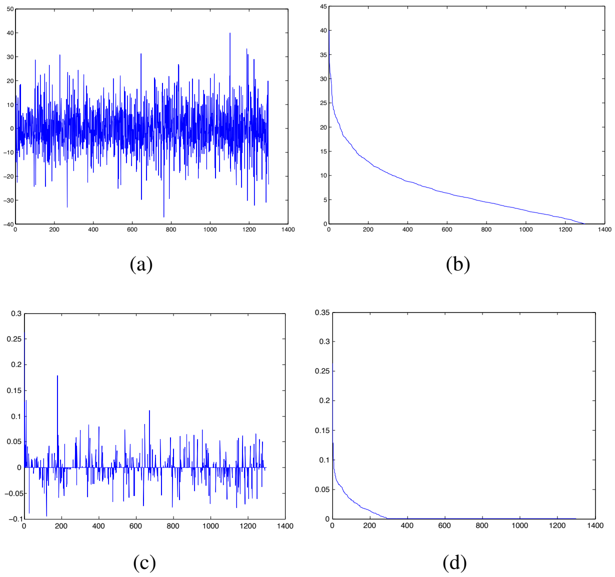
Having seen that the the training data are not linearly dependent we now show that the sparsity espoused in [19, 18] and [14] is not a feature of the problem, but of the solution. In applying the matrix $\mathbf{\Phi} \in \mathbb{R}^{d \times m}$ (where $d \ll m$) the methods proposed cause the problem to become sparse, with a degree proportional to the value of d selected. Figure 2 shows the values of $\boldsymbol{\alpha}$ estimated by solving equation (4) directly when $\mathbf{A}$ is constructed as above, but from 13 images each of 100 subjects, and $\mathbf{x}$ represents another image from the AR dataset. Two matrices $\mathbf{\Phi}$ have been used, one with $d = n - 1$ and one with $d = 300$. The plots show that the coefficients $\boldsymbol{\alpha}$ are not sparse until the selection of a small d forces them to be so.

We show below that the CS methods for face recognition listed above achieve their state-of-the-art results on the AR and Yale B datasets only when the the number of features d is at least 300. This and Figure 2 imply that the coefficients of $\boldsymbol{\alpha}$ are not as sparse as may have been hoped, and that at least 300 non-zero coefficients are required in order to achieve acceptable classification performance.

This analysis draws into question the theoretical support for all face recognition methods based on Assumption 1 and any method relying on the sparsity of the coefficients $\boldsymbol{\alpha}$. This does not mean that the ℓ_1 norm has no place in face recognition, however, but rather that it needs to be applied appropriately.

3. Robust vs. sparse ℓ_1

One argument with the analysis above might be that the ℓ_1 term is intended to achieve robustness, rather than indicating a belief in the sparsity of the coefficients. This is an important distinction. The ℓ_1 norm is used in CS as a tractable alternative to the ℓ_0 norm [3]. Sparseness does not



(a)

(b)

(c)

(d)

Figure 2. Visualising the sparsity of $\boldsymbol{\alpha}$ when recovered by solving equation (4). (a) Plot of $\boldsymbol{\alpha}$ when $d = 1299$. (b) Plot of sorted $\boldsymbol{\alpha}$ when $d = 1299$. (c) Plot of $\boldsymbol{\alpha}$ when $d = 300$. (d) Plot of sorted $\boldsymbol{\alpha}$ when $d = 300$.

necessarily lead to robustness to the presence of outliers.

To achieve robustness, one could use ℓ_1-Regression [16, chap. 12.4] as follows:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\alpha}\|_{\ell_1}, \tag{6}$$

which avoids overly penalising gross outliers.

The ℓ_1 norm in CS and ℓ_1 norm in ℓ_1-Regression are, however, two unrelated uses of the same norm. The two applications differ in the quantities to which the ℓ_1 norm is applied. Robustness cannot therefore be used to justify applying the ℓ_1-norm to the coefficients $\boldsymbol{\alpha}$ without further explanation. No such explanation is given, however.

The problem with (6) is that solving such a linear program can be computationally expensive as the data size grows. Rather than resort to approaches such as those in (4) and (5), however, we now show that faster, more accurate, and more robust methods may be achieved by modelling outliers explicitly and using the ℓ_2 norm.

4. The orthonormal ℓ_2 norm method

In contrast to the ℓ_1 case, it is possible to estimate $\boldsymbol{\alpha}$ using the ℓ_2 norm by solving

$$\operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^n} \|\mathbf{x} - \mathbf{A}\boldsymbol{\alpha}\|^2_{\ell_2}. \tag{7}$$

Even when the system is overdetermined, the optimal solution, (in the sense of the smallest reconstruction error) can be recovered by $\boldsymbol{\alpha} = (\mathbf{A^T A})^{-1}\mathbf{A^T}\,\mathbf{x}$.

Algorithm 1 Orthonormal ℓ_2 Face Recognition

```
**Input:** a training image matrix $\mathbf{A}$ for $K$ subjects, a test image matrix $\mathbf{X}$.
Compute $\mathbf{QR} = \mathbf{A}$.
**for** $\mathbf{x} \in \mathbf{X}$ **do**
  $\boldsymbol{\alpha} = \mathbf{R^{-1} Q^T}\,\mathbf{x}$
  find the identity of image $x$ via (10).
**end for**
**Output:** identity for all test images.
```

Solving (7) efficiently requires re-formulating the psuedo-inverse, however. By QR factorisation of $\mathbf{A}$, we have $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ forms a orthonormal basis, and $\mathbf{R}$ is an upper triangle matrix. Therefore,

$$(\mathbf{A^T A})^{-1}\mathbf{A^T} = \mathbf{R^{-1} Q^T}. \tag{8}$$

Consequently, we can estimate α via

$$\boldsymbol{\alpha} = \mathbf{R^{-1} Q^T}\,\mathbf{x}\,. \tag{9}$$

Here $\mathbf{R^{-1} Q^T}$ remains the same for all $\mathbf{x}$. So we just need to compute $\mathbf{R}^{-1}\mathbf{Q^T}$ once and store it. If a set of test images is provided as $\mathbf{X}$ whose columns are test images, then $\Lambda = \mathbf{R^{-1} Q^T}\,\mathbf{X}$.

Once the coefficients are estimated, one can find the identity of image x via minimising the residuals

$$c^*(\mathbf{x}) = \operatorname*{argmin}_k ||\mathbf{x} - \mathbf{A}_k\boldsymbol{\alpha}_k||_{\ell_2} \tag{10}$$

for $k = 1, \ldots, K$, where $\boldsymbol{\alpha}_k$ is the n_k dimensional subvector consisting of components of $\boldsymbol{\alpha}$ and $\mathbf{A}_k$ is a m-by-n_k submatrix of $\mathbf{A}$, both corresponding to the basis of person k. A similar reformulation applies to the Nearest Subspace method [6].

It should also be noted that the distance measure used in (10) is different from that of the nearest subspace method. This difference becomes apparent when comparing the coordinates given by the respective approaches:

$$\boldsymbol{\alpha}_k = \mathbf{I}_k \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{x} \tag{11}$$

$$\boldsymbol{\alpha}_k^{NS} = \left(\mathbf{A}_k^T\mathbf{A}_k\right)^{-1}\mathbf{A}_k^T\mathbf{x}, \tag{12}$$

where $\mathbf{I}_k = \begin{bmatrix} 0 & \ldots & I & \ldots & 0 \end{bmatrix}$ is a matrix extracting the coordinates corresponding to the k-th individual from $\boldsymbol{\alpha}$. These differences are discussed in length in the supplementary material of [18].

5. Face recognition without corruption

The above orthonormal ℓ_2 minimisation approach (Algorithm 1), leads to a very efficient face recognition method when faces are not corrupted by random noise or foreign objects.

In order to compare to CS based face recognition, we use the same datasets (Extended YaleB and AR) as [18, 19, 14].

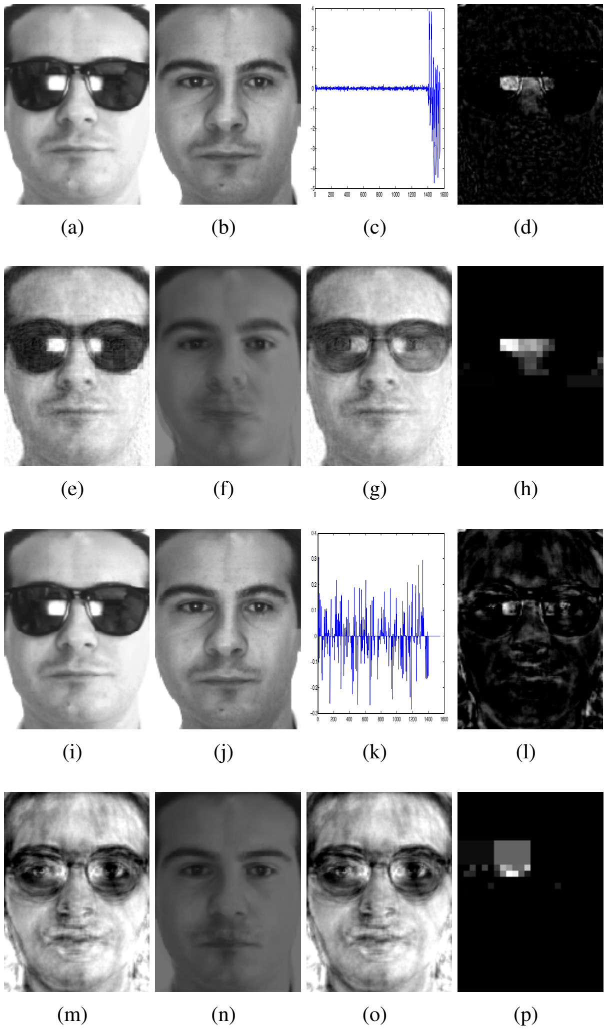


Figure 3. Prediction on face wearing sunglasses. **Top two rows:** for ℓ_2 method. **Bottom two rows:** for ℓ_1 method. (a) and (i) are the test faces. (b) and (j) are used to show the person's identity by displaying the first training image from that person. (c) and (k) are the plots of the estimated $\hat{\boldsymbol{\alpha}}$. (d) and (l) are the gross residual images $\mathbf{x} - [\mathbf{A}, \mathbf{B}]\hat{\boldsymbol{\alpha}}$. (e) and (m) are the reconstructed images by $\hat{\boldsymbol{\alpha}}$, *i.e.* $\mathbf{A}\boldsymbol{\alpha} + \mathbf{B}\boldsymbol{\beta}$. (f) and (n) are the reconstructed images by the predicted person's training images only, *i.e.* $\mathbf{A}\boldsymbol{\alpha}^{c^*}$. (g) and (o) are the reconstructed images by all subjects *i.e.* $\mathbf{A}\boldsymbol{\alpha}$. (h) and (p) are $\mathbf{B}\boldsymbol{\beta}$. For visualisation purpose, all residual images (d,h,l,p) are re-scaled such that the highest pixel value is 255.

Datasets The AR dataset consists of over $3,000$ frontal images of 126 individuals. There are 26 images of each individual, taken at two different occasions [10]. The faces in AR contain variations such as illumination change, expressions and facial disguises (*i.e.* sun glasses or scarf). We randomly selected 100 subjects (50 male and 50 female) for our experiments. For each subject, we randomly permute the 26 images and then take the first half for training and the rest for testing. This way, we have $1,300$ images for training and $1,300$ images for testing. For statistical stability, we generate 10 different training and test dataset pairs by randomly permuting 10 times. The extended YaleB dataset [7] consists of $2,414$ frontal face images of 38 subjects. They are captured under various lighting conditions and cropped and normalised to 192×168 pixels. For our experiment, we take 62 images per person thus in total we use $2,356$ images. Again for each subject, we randomly permute the 62 images and take the first half for training and the rest for testing. For statistical stability, we generate 10 different training and test dataset pairs.

Performance comparisons We compare our fast ℓ_2 norm method to the ℓ_1 norm method with a Gaussian random matrix [18], OMP on a Gaussian random matrix and OMP on a Hash matrix [14] and the Nearest Subspace method [6]. Wright *et al.* in [18] use flattened raw pixel values as features after downsampling the images claiming that this is necessary for computational tractability. However, we show our ℓ_2 norm method has no problem dealing with the original feature dimension. We thus simply flatten the original 165×120 images to feature vectors of length 19800.

All of the methods listed above need to estimate $\boldsymbol{\alpha}$, then check the test image's identity by identifying the minimal residual. There are also off-line processes (independent of the test images) which need to be carried out for each of the methods. For the ℓ_2 norm method, $\mathbf{R^{-1}Q^T}$ must be computed, but this can happen before hand. For the ℓ_1 norm method, the Gaussian random matrix $\boldsymbol{\Phi}$ must be generated and $\boldsymbol{\Phi}\mathbf{A}$ computed. For both OMP methods (Random OMP and Hash OMP), each column of $\mathbf{A}$ must be normalied to unit length (ℓ_2 distance). Random OMP requires a Gaussian random matrix $\boldsymbol{\Phi}$ and that $\boldsymbol{\Phi}\mathbf{A}$ be computed. In principle, Hash OMP does not need to compute the Hash matrix $\mathbf{H}$ explicitly. One can feed the data stream into the hash code and generate the $\mathbf{HA}$ on the fly. However, for ease of comparison, we generate it explicitly here and compute $\mathbf{HA}$. We solve (4) using `CVX`, a package for specifying and solving convex programs [9, 8].

Results All algorithms are evaluated on the training and test dataset pairs constructed as described above. The comparison results for the AR dataset are reported in Table 1. As we can see, ℓ_2 has the highest average recognition rate at 95.89%, and the second best is the ℓ_1 with norm $d = 300$ which acheives 93.12%. What is notable here is that ℓ_2 takes only 2.71 seconds (in matlab) to estimate $\boldsymbol{\alpha}$ for all $1,300$ test images, which is over $2,000$ times faster than the ℓ_1 method with 2.77% higher recognition rate. The offline process of the ℓ_2 norm method takes only 28.74 seconds, which is negligible for an offline process. The speed of the Nearest Subspace method is comparable with that of the ℓ_2 norm method, but it has it has worse average recognition rate and significantly larger standard deviation. Likewise,

Table 5. CheckID running time (for the ℓ_2 norm method) vs. recognition rate. d = the number of rows in $\mathbf{\Phi}$. Time = running time for check faces' identification. All running time are in seconds. RecRate = Recognition Rate. Data were randomly permuted 10 times, thus all measures are reported as the average ± standard deviation.

d	Time	RecRate
2^{13}	28.14 ± 0.60s	95.90 ± 2.35%
2^{11}	7.34 ± 0.02s	95.92 ± 2.33%
2^{9}	2.47 ± 0.03s	95.90 ± 2.35%
2^{7}	1.05 ± 0.03s	95.87 ± 2.33%
2^{5}	0.90 ± 0.03s	95.88 ± 2.21%
2^{3}	0.88 ± 0.03s	95.57 ± 2.42%
2^{2}	0.80 ± 0.01s	95.08 ± 2.65%
2^{1}	0.79 ± 0.01s	85.35 ± 1.98%

on the YaleB dataset, the ℓ_2 norm method outperforms its competitors as shown in Table 2. We do not report the results for d which lead to non-competitive results in Table 2. All experiments are conducted in Matlab running on a PC with a 2.8GHz CPU with 8GB Memory.

Improving CheckID performance Estimating the coefficients using the ℓ_2-based method is so fast that the time taken to check the identity of the result (CheckID) is the dominating factor in its execution time. We can, however, improve the speed of the CheckID process without noticeably degrading the recognition rate. Recall that CheckID uses (10) for all methods except the nearest subspace method (reported in Table 1 and 2). Instead, given an estimated $\boldsymbol{\alpha}$, we can check the identity in a random projected space, that is $\hat{c}^*(\mathbf{x}) = \mathrm{argmin}_k \, ||\mathbf{\Phi x} - \mathbf{\Phi A}_k \boldsymbol{\alpha}_k||_{\ell_2}$, where $\mathbf{\Phi} \in \mathbb{R}^{d,m}$. Note that if the test image set denoted as $\mathbf{A}_{test}$ is known, then $\mathbf{\Phi A}_{test}$ and $\mathbf{\Phi A}$ only need to be computed once and the complexity of CheckID decreases as d decreases. Fortunately, we discover that decreasing d significantly speeds up the CheckID without noticeably degenerating the recognition rate, as shown in Table 5. For example, CheckID of ℓ_2 norm method takes 69.15 seconds in Table 1 with recognition rate 95.89%, whereas it takes only 0.88 seconds with recognition rate 95.57% when $d = 2^3$. In fact, the recognition rate only has a significant drop at $d = 2^1$. The recognition rate is preserved despite the small values of d in the spirit of Johnson-Lindenstrauss Lemma [1].

6. Face recognition with corruption

White noise is quite common, and commonly assumed in signal transmission problems. We therefore add random noise from normal distributions to the existing AR images in order to test resilience to normal noise sources. That is,

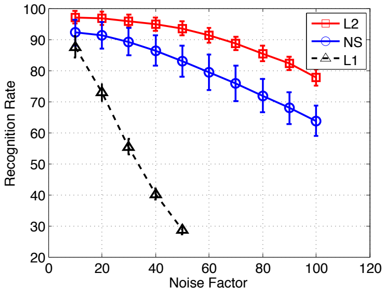


Figure 4. Recognition Rate v.s. Noise Factor on the AR dataset images with additive Gaussian noise.

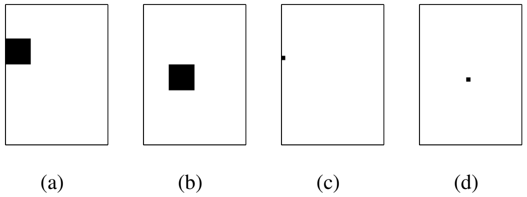


Figure 5. Boxes representing the disguising objects. The boxes have intensity 0.1×255 and non-box areas have intensity 0.

$$\acute{\mathbf{x}}_i = (\mathbf{x}_i + b\,\mathbf{z}_i), \quad i = 1, \ldots n$$

where $\mathbf{z}_i \sim \mathcal{N}(0, 1)$, and $b > 0$ is the noise factor.

In practice, we need to make sure $\acute{\mathbf{x}}_i$ is still a valid 8-bit grey scale image, which can be simply done by truncating pixel values outside the interval $[0,\ 255]$ before applying all methods to the noisy dataset. We test ℓ_2, ℓ_1 based methods and the Nearest Subspace on images with noise. We are not interested in testing Random OMP and Hash OMP here because (1) they are just fast greedy methods for looking for sparse solutions. In terms of the precision on sparse signal recovery, ℓ_1 is superior to them. (2) testing all of them on 10 different noise factor with 10 different data split takes too much running time. Figure 4 shows that the recognition rate for the ℓ_2 norm method's is preserved reasonably well as the noise factor b increases. The Nearest Subspace method performs second best with nearly double the standard deviation (shown as the error bar width). The ℓ_1 norm method performs poorly as the noise factor increases[3] This suggests that sparseness reinforcement on the $\boldsymbol{\alpha}$ does not necessarily lead to robustness.

7. Face recognition with disguise

Now we study how the ℓ_2 norm method and the competitors perform on faces with disguise. In the AR dataset, there are 26 images of each person : 14 images without disguises

[3] We did not evaluate the ℓ_1 norm method for $b > 50$ as its speed and accuracy had diminished so far that it was not warranted.

Table 1. Recognition Rate and Running Time on AR dataset. Offline = running time for offline processing. Est = running time for estimating coefficients. CheckID = running time for checking face identification for all test images (not per image). All running time are in seconds. RecRate = Recognition Rate. Data were randomly permuted 10 times, thus all measures are reported as the average ± standard deviation. Lowest running time and highest recognition rate are in bold.

Algorithms	**Offline**	**Est**	**CheckID**	**RecRate**
ℓ_2	28.74 ± 0.37s	**2.71 ± 0.02**s	69.15 ± 0.32s	**95.89 ± 2.35**%
$\ell_1(d = 300)$	1.01 ± 0.01s	5519.01 ± 23.70s	91.20 ± 0.77s	93.12 ± 2.94%
$\ell_1(d = 200)$	0.68 ± 0.01s	2893.47 ± 67.41s	102.16 ± 1.79s	91.54 ± 3.15%
$\ell_1(d = 100)$	**0.35 ± 0.00**s	1068.20 ± 25.94s	102.13 ± 1.50s	86.13 ± 3.87%
Random OMP$(d = 300)$	1.98 ± 0.01s	1177.52 ± 3.02s	91.90 ± 0.28s	84.85 ± 3.43%
Random OMP$(d = 200)$	1.64 ± 0.02s	348.88 ± 1.24s	85.75 ± 0.15s	80.52 ± 4.12%
Random OMP$(d = 100)$	1.31 ± 0.01s	44.85 ± 0.78s	60.95 ± 0.12s	64.68 ± 5.50%
Hash OMP$(d = 300)$	4.51 ± 0.04s	153.08 ± 7.39s	63.90 ± 0.94s	86.92 ± 3.44%
Hash OMP$(d = 200)$	4.21 ± 0.02s	38.37 ± 2.11s	59.90 ± 0.39s	82.99 ± 3.63%
Hash OMP$(d = 100)$	3.93 ± 0.01s	7.05 ± 0.20s	58.33 ± 0.11s	64.49 ± 5.27%
Nearest Subspace	1.06 ± 0.06s	3.07 ± 0.03s	**0.07 ± 0.01**s	92.32 ± 4.16%

Table 2. Recognition Rate and Running Time on YaleB dataset.

Algorithms	**Offline**	**Est**	**CheckID**	**RecRate**
ℓ_2	29.02 ± 0.25s	**3.55 ± 0.09**s	70.60 ± 0.71s	**98.91 ± 1.37**%
$\ell_1(d = 300)$	**1.52 ± 0.01**s	4191.34 ± 14.16s	79.48 ± 0.03s	96.63 ± 3.03%
Random OMP$(d = 200)$	2.43 ± 0.07s	12291.77 ± 87.31s	48.21 ± 0.19s	93.75 ± 4.40%
Hash OMP $(d = 300)$	7.04 ± 0.09s	3246.28 ± 250.37s	51.09 ± 0.98s	94.92 ± 3.86%
Nearest Subspace	2.74 ± 0.03s	3.83 ± 0.04s	**0.02 ± 0.00**s	96.87 ± 2.12%

but with various facial expressions and illumination conditions, 6 images with sunglasses and 6 images with scarves. We thus split the dataset into a training set (*i.e.* 1400 unoccluded faces only), a sunglasses test set (600 images of subjects wearing sunglasses) and a scarves test set (600 images of subjects wearing scarves). This ensures that none of the disguising objects (sunglasses or scarves) appears in the training set. Note that in [18] *only a subset (200 out of 600) of disguised images are used for testing in each disguise case*. When we apply all competitors to the full test sets, the results are very different from what was reported there, which will be discussed in detail later in this section after we introduce a method for dealing with the disguising objects.

To represent the disguising objects Wright *et al.* in [18] expand the basis by a square identity matrix $\mathbf{I}$, then seek $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by the following ℓ_1 minimisation:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n,\boldsymbol{\beta}\in\mathbb{R}^m} \|[\begin{array}{c}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{array}]\|_{\ell_1} \tag{13a}$$

$$\text{s.t. } \boldsymbol{\Phi}\mathbf{x} = \boldsymbol{\Phi}[\mathbf{A},\mathbf{I}][\begin{array}{c}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{array}], \tag{13b}$$

or alternatively

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n,\boldsymbol{\beta}\in\mathbb{R}^m} \|[\begin{array}{c}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{array}]\|_{\ell_1} \tag{14a}$$

$$\text{s.t. } \|\boldsymbol{\Phi}\mathbf{x} - \boldsymbol{\Phi}[\mathbf{A},\mathbf{I}][\begin{array}{c}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{array}]\|_{\boldsymbol{\ell_2}} \leq \varepsilon. \tag{14b}$$

Identity is again determined by identifying the minimal residuals among all subjects. This is problematic, however, since $\mathbf{I}$ can represent any possible face image without $\mathbf{A}$. Alternatively, they construct more sophisticated features (*e.g.* partition features) to improve the performance of the ℓ_1 norm method. However, the features are not applied to other competitors in [18], thus it is not clear that whether the improvement comes from the ℓ_1 norm method or purely from the new features.

We use a similar method (but with significantly fewer additional columns) to cope with the disguise. The key idea is to try to let $\boldsymbol{\beta}$ only represent non-face objects and let $\boldsymbol{\alpha}$ only represent faces. Clearly an identity $\mathbf{I}$ is not a good choice for it is able to represent any image with that size. Thus we generate a number of images with one grey box in various locations to represent reasonable size objects. In the experiment, we use 8 large (30 by 30) box images and 144 small (5 by 5) box images[4] shown in Figure 5. The face

[4] In fact, users can design other images as long as the images follow the "key idea" mentioned above.

Table 3. Performance comparison when subjects are disguised with sunglasses and scarves. Since we use all non-disguised faces as training set, we have one unique data split.

	Wearing Sunglasses				Wearing Scarves			
Algorithms	**Offline**	**Est**	**CheckID**	**RecRate**	**Offline**	**Est**	**CheckID**	**RecRate**
ℓ_2	48.22s	**1.49s**	35.01s	**78.50%**	47.52s	**1.50s**	34.89s	**79.50%**
$\ell_1(d = 300)$	0.87s	2917.69s	47.56s	40.17%	0.93s	2935.33s	47.37s	55.17%
Random OMP$(d = 300)$	1.60s	426.05s	40.20s	43.00%	1.70s	3170.02s	39.75s	27.00%
Hash OMP$(d = 300)$	4.12s	189.95s	38.66s	46.50%	4.20s	1660.65s	37.29s	32.50%

Table 4. ℓ_1 results on the downsampled AR dataset with disguise. Correct = the number of correct predictions of the test images. ℓ_1 = LP form uses (13). ℓ_1r = the reduced problem uses (14) . Both use $d = 300$. m is the size of images after downsampling.

	Wearing Sunglasses					Wearing Scarves				
Algorithms	**Offline**	**Est**	**CheckID**	**Correct**	**RecRate**	**Offline**	**Est**	**CheckID**	**Correct**	**RecRate**
$\ell_1(m = 540)$	0.19s	3679.03s	2.78s	294	49.00%	0.20s	3738.69s	2.80s	378	63.00%
$\ell_1(m = 130)$	0.05s	2903.38s	0.97s	220	36.67%	0.05s	2853.37s	0.96s	179	29.83%
ℓ_1r$(m = 540)$	0.19s	4828.12s	2.80s	291	48.50%	0.20s	4740.44s	2.78s	378	63.00%
ℓ_1r$(m = 130)$	0.05s	4156.75s	0.98s	220	36.67%	0.06s	4148.06s	1.00s	180	30.00%

images and the box images can be downloaded from the authors' website. Stacking the box images as columns, we get a matrix $\mathbf{B}$. Let $\hat{\mathbf{A}} = [\mathbf{A}, \mathbf{B}]$ and $\hat{\boldsymbol{\alpha}} = [\begin{smallmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{smallmatrix}]$, and then input $\hat{\mathbf{A}}$ (instead of $\mathbf{A}$) to Algorithm 1 to estimate $\hat{\boldsymbol{\alpha}}$ (instead of $\boldsymbol{\alpha}$). The person id is predicted via minimal residuals over all $\boldsymbol{\alpha}^{\boldsymbol{k}}$, while $\boldsymbol{\beta}$ can be ignored as it is shared by all subjects to represent the disguising objects.

In order to ensure a fair comparison all competitors have been tested using the same $\hat{\mathbf{A}}$. Since we use all non-disguised faces as the training set, we have one unique data split. Table 3 shows that in the case of both sunglasses and scarves the ℓ_2 norm method outperforms its competitors by a very large margin in terms of recognition rate and estimation running time. In particular, ℓ_2 achieves 38.33% higher recognition rate than ℓ_1 in sunglasses case and 24.33% higher recognition rate than ℓ_1 in scarves case with over 2,000 times speed up.

It is interesting to note that the image reconstructed by the ℓ_1 coefficients is highly distorted (Figure 3(m)) whereas that reconstructed by ℓ_2 (Figure3(e)) is more faithful to the original image. The ℓ_1 norm gives a sparse $\boldsymbol{\alpha}$ whereas ℓ_2 norm gives a dense one as expected (see Figure 3(k) and 3(c)). However, a sparse $\boldsymbol{\alpha}$ does not necessarily lead to a more robust estimation. In fact, from Table 3, the dense $\boldsymbol{\alpha}$ via the ℓ_2 norm outperforms the sparse one via ℓ_1 in recognition rate by a significant amount.

We also tested the Nearest Subspace method on this dataset. Since the projection onto the additional $\mathbf{B}$ is not meaningful we instead used $\mathbf{A}$, and achieved a recognition rate of 62.83% on the sunglasses test set and 13.83% recognition rate on the scarves test set. The result is not directly comparable to those in Table 3, but is still informative.

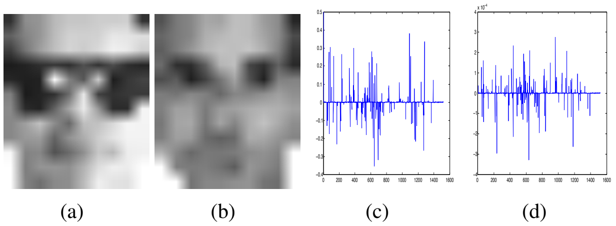


Figure 6. Prediction on downsampled AR faces with size 13 × 10. (a) test face. (b) predicted test face by ℓ_1 and ℓ_1r. (c) estimated coefficient $\boldsymbol{\alpha}$ via ℓ_1. The coefficient achieved by ℓ_1r is very similar to (c), thus it is not presented here. (d) The difference of the estimated coefficients by ℓ_1 and ℓ_1r. The difference is only in $O(10^{-4})$.

Performance comparison against Wright *et al.* [18] In [18] the downsampling of AR face images from 165 × 120 to 27 × 20 and 13 × 10 is justified as being necessary for computational tractability. They train on 799 unoccluded images and test on two separate test sets (*i.e.* sunglasses and scarves) of 200 images. Since it is not stated which 799 of the 1400 unoccluded images or which 200 of the 600 sunglasses(or scarves) images are used, we have selected all 1,400 unoccluded images as the training set, and 600 sunglasses images and 600 scarf images as two separate testing sets. To better compare with their results, we downsample AR images to 27 × 20 and 13 × 10 as well, though the downsampling step itself is arguable: after downsampling , the 13 × 10 images are hardly recognisable as faces and it is extremely difficult for a human to recognise the subjects' identities (see Figure 6 (a) and (b)). We use both (13) and (14) as in [18]. Here (13) is a linear program, hence it is expected to be faster than (14), which is a convex problem (a second-order cone program) [5]. We solve both problems using CVX [9, 8]. For (14), we set $\varepsilon = 0.05$ as in [18]. The results are reported in Table 4. Both (13) and (14) produce almost identical recognition rates though (13) is faster as expected. The difference between the estimated coefficients is very small (in $O(10^{-4})$ see Figure 6(d)). Comparing to Table 3, downsampled ℓ_1 still produces results inferior to ℓ_2. Moreover, the recognition rate of downsampled ℓ_1 decreases as the image size m decreases.

8. Discussion

In this work we have compared Compressive Sensing face recognition methods, such as [18] and [14], with standard ℓ_2 approaches. The conclusion we have drawn as a result is that there is no theoretical or empirical reason to expect that enforcing sparsity on the coefficients of (2) will improve robustness. The experiments carried out here clearly demonstrate this. Not only does solving (4) lead to worse performance, it is also less robust and orders of magnitudes slower than least-squares type approaches.

We do not propose a novel robust method for face recognition, but rather show that well know least-squares approaches out perform many of the existing more complicated algorithms. We also showed that if ℓ_1 minimisation is intended to improve the robustness of the method then this should be achieved by solving (6) as discussed in section 3. This may be computationally expensive, however, as it requires solving a linear program. Ways of efficiently solving (6) and an investigation in to the performance of such a formulation is the topic of future work.

References

[1] R.G. Baraniuk, M. Davenport, R. DeVore, and M.B. Wakin. A simple proof of the restricted isometry principle for random matrices. *Constructive Approximation*, 2007.

[2] R. Basri and D.W. Jacobs. Lambertian reflectance and linear subspaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):218 – 233, February 2003.

[3] E. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Information Theory*, 52(2):489–509, 2006.

[4] E. Candés and T. Tao. Decoding by linear programming. *IEEE Trans. Information Theory*, 51(12):4203–4215, 2005.

[5] E.J. Candes and M.B. Wakin. An introduction to compressive sensing. *IEEE Signal Processing Magazine*, pages 21–30, 2008.

[6] Kuang chih Lee, Jeffrey Ho, and David Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:684–698, 2005.

[7] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.

[8] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[9] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, October 2010.

[10] A. Martinez and R. Benavente. The ar face database. Technical Report 24, CVC Tech. Report, 1998.

[11] M. Rudelson and R. Veshynin. Geometric approach to error correcting codes and reconstruction of signals. *Int. Math. Res. Notices*, 64:4019–4041, 2005.

[12] Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, A. Strehl, and S. V. N. Vishwanathan. Hash kernels. In *Proc. Int. Workshop Artificial Intell. & Statistics*, 2009.

[13] Q. Shi, J. Petterson, G. Dror, J. Langford, A. J. Smola, and S.V.N. Vishwanathan. Hash kernels for structured data. *J. Mach. Learn. Res.*, 10:2615–2637, 2009.

[14] Qinfeng Shi, Hanxi Li, and Chunhua Shen. Rapid face recognition using hashing. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, USA, 2010.

[15] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Information Theory*, 53(12):4655–4666, 2007.

[16] Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. 2nd, edition, 2008.

[17] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A.J. Smola. Feature hashing for large scale multitask learning. In L. Bottou and M. Littman, editors, *Proc. Int. Conf. Mach. Learn.*, 2009.

[18] J. Wright, A. Y. Yang, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intelli.*, 2008.

[19] A. Y. Yang, J. Wright, Y. Ma, and S. S. Sastry. Feature selection in face recognition: A sparse representation perspective. *Tech. Report*, 2007.