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 $\ell_2$ 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 $\ell_2$-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 $A$ with $m$ rows and $n$ columns, given by

$$A = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in \mathbb{R}^{m \times n}. \quad (1)$$

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 $x$, lying in the subspace spanned by the training images belonging to the same person. That is for any test image $x$, without knowing its label information, it’s assumed that there exists a $\eta$-sparse $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ such that

$$x = A\alpha. \quad (2)$$

To seek a sparse solution, one could use

$$\min_{\alpha \in \mathbb{R}^n} \|\alpha\|_1 \quad (3a)$$

s.t. $x = A\alpha. \quad (3b)$

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 $\Phi \in \mathbb{R}^{d \times m}$ (where $d \ll m$) and identify the vector $\alpha$ which minimises the following $\ell_1$ problem:

$$\min_{\alpha \in \mathbb{R}^n} \|\alpha\|_1 \quad (4a)$$

s.t. $\Phi x = \Phi A \alpha. \quad (4b)$

or the related problem:

$$\min_{\alpha \in \mathbb{R}^n} \|\alpha\|_1 \quad (5a)$$

s.t. $\|\Phi x - \Phi A \alpha\|_2 \leq \varepsilon. \quad (5b)$

---

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 $\eta$-sparse if it has at most $\eta$ non-zero entries.
for a given error tolerance $\varepsilon > 0$. Introducing the matrix $\Phi$ 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 $\alpha$ 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 $\Phi$ with an implicit hash matrix $H$ 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 $A$ 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 $\times$ 120 pixels and converted to grayscale (as in [19]), and, using (1), formed the matrix $A$ 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 $A$ 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 $A$ 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 $A$ 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.

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 $\Phi \in \mathbb{R}^{d \times m}$ (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 $\alpha$ estimated by solving equation (4) directly when $A$ is constructed as above, but from 13 images each of 100 subjects, and $x$ represents another image from the AR dataset. Two matrices $\Phi$ have been used, one with $d = n - 1$ and one with $d = 300$. The plots show that the coefficients $\alpha$ 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 $\alpha$ 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 $\alpha$. This does not mean that the $\ell_1$ norm has no place in face recognition, however, but rather that it needs to be applied appropriately.

3. Robust vs. sparse $\ell_1$

One argument with the analysis above might be that the $\ell_1$ term is intended to achieve robustness, rather than indicating a belief in the sparsity of the coefficients. This is an important distinction. The $\ell_1$ norm is used in CS as a tractable alternative to the $\ell_0$ norm [3]. Sparseness does not
necessarily lead to robustness to the presence of outliers.

To achieve robustness, one could use \( \ell_1 \)-Regression [16, chap. 12.4] as follows:

\[
\min_{\alpha \in \mathbb{R}^n} \| \mathbf{x} - \mathbf{A}\alpha \|_{\ell_1},
\]

which avoids overly penalising gross outliers.

The \( \ell_1 \) norm in CS and \( \ell_1 \) norm in \( \ell_1 \)-Regression are, however, two unrelated uses of the same norm. The two applications differ in the quantities to which the \( \ell_1 \) norm is applied. Robustness cannot therefore be used to justify applying the \( \ell_1 \)-norm to the coefficients \( \alpha \) 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 \( \ell_2 \) norm.

4. The orthonormal \( \ell_2 \) norm method

In contrast to the \( \ell_1 \) case, it is possible to estimate \( \alpha \) using the \( \ell_2 \) norm by solving

\[
\text{argmin}_{\alpha \in \mathbb{R}^n} \| \mathbf{x} - \mathbf{A}\alpha \|_{\ell_2}.
\]

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

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

\[
(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = \mathbf{R}^{-1} \mathbf{Q}^T.
\]

Consequently, we can estimate \( \alpha \) via

\[
\alpha = \mathbf{R}^{-1} \mathbf{Q}^T \mathbf{x}.
\]

Here \( \mathbf{R}^{-1} \mathbf{Q}^T \) 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

\[
\mathbf{A} = \mathbf{R}^{-1} \mathbf{Q}^T \mathbf{X}.
\]

Once the coefficients are estimated, one can find the identity of image \( \mathbf{x} \) via minimising the residuals

\[
\mathbf{e}^*(\mathbf{x}) = \text{argmin}_{k} \| \mathbf{x} - \mathbf{A}_k \alpha_k \|_{\ell_2}
\]

for \( k = 1, \ldots, K \), where \( \alpha_k \) is the \( n_k \) dimensional subvector consisting of components of \( \alpha \) and \( \mathbf{A}_k \) is a \( m \times n_k \) 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:

\[
\alpha_k = \mathbf{I}_k (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{x}
\]

\[
\alpha_k^{NS} = (\mathbf{A}_k^T \mathbf{A}_k)^{-1} \mathbf{A}_k^T \mathbf{x},
\]

where \( \mathbf{I}_k = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \end{bmatrix} \) is a matrix extracting the coordinates corresponding to the \( k \)-th individual from \( \alpha \). These differences are discussed in length in the supplementary material of [18].

5. Face recognition without corruption

The above orthonormal \( \ell_2 \) 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].

\begin{algorithm}
\caption{Orthonormal \( \ell_2 \) Face Recognition}
\begin{algorithmic}
\STATE \textbf{Input:} a training image matrix \( \mathbf{A} \) for \( K \) subjects, a test image matrix \( \mathbf{X} \).
\STATE Compute \( \mathbf{Q} \mathbf{R} = \mathbf{A} \).
\FOR{\( x \in \mathbf{X} \)}
\STATE \( \alpha = \mathbf{R}^{-1} \mathbf{Q}^T \mathbf{x} \)
\STATE find the identity of image \( x \) via (10).
\ENDFOR
\STATE \textbf{Output:} identity for all test images.
\end{algorithmic}
\end{algorithm}
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. sunglasses 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 $\ell_2$ norm method to the $\ell_1$ 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 $\ell_2$ 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 $\alpha$, 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 $\ell_2$ norm method, $R^{-1}Q^T$ must be computed, but this can happen before hand. For the $\ell_1$ norm method, the Gaussian random matrix $R$ must be generated and $RA$ computed. For both OMP methods (Random OMP and Hash OMP), each column of $A$ must be normalised to unit length ($\ell_2$ distance). Random OMP requires a Gaussian random matrix $R$ and that $RA$ be computed. In principle, Hash OMP does not need to compute the Hash matrix $H$ explicitly. One can feed the data stream into the hash code and generate the $HA$ on the fly. However, for ease of comparison, we generate it explicitly here and compute $HA$. 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, $\ell_2$ has the highest average recognition rate at 95.89%, which achieves 93.12%. What is notable here is that $\ell_2$ takes only 2.71 seconds (in matlab) to estimate $\alpha$ for all 1,300 test images, which is over 2,000 times faster than the $\ell_1$ method with 2.77% higher recognition rate. The offline process of the $\ell_2$ 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 $\ell_2$ norm method, but it has it has worse average recognition rate and significantly larger standard deviation. Likewise,
on the YaleB dataset, the $\ell_2$ 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 $\ell_2$-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 $\alpha$, we can check the identity in a random projected space, that is

$$c^* (x) = \arg \min_{k} ||\Phi x - \Phi A_k \alpha_k||_{\ell_2},$$

where $\Phi \in \mathbb{R}^{d \times m}$. Note that if the test image set denoted as $A_{test}$ is known, then $\Phi A_{test}$ and $\Phi A$ 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 de-generating the recognition rate, as shown in Table 5. For example, CheckID of $\ell_2$ norm method takes 69.15 seconds in Table 1 with recognition rate 95.89%, 0.88 seconds with recognition rate 95.57% when $d = 2^5$. 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,

$$\tilde{x}_i = (x_i + b z_i), \quad i = 1, \ldots, n,$$

where $z_i \sim N(0, 1)$.

In practice, we need to make sure $\tilde{x}_i$ 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 $\ell_2$, $\ell_1$ 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, $\ell_1$ 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 $\ell_2$ 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 $\ell_1$ norm method performs poorly as the noise factor increases. This suggests that sparseness reinforcement on the $\alpha$ does not necessarily lead to robustness.

### 7. Face recognition with disguise

Now we study how the $\ell_2$ 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.

---

We did not evaluate the $\ell_1$ norm method for $b > 50$ as its speed and accuracy had diminished so far that it was not warranted.
we thus split the dataset into a training set. Note that in [18] there, which will be discussed in detail later in this section after we introduce a method for dealing with the disguising objects.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Offline</th>
<th>Est</th>
<th>CheckID</th>
<th>RecRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_2$</td>
<td>28.74 ± 0.37s</td>
<td><strong>2.71 ± 0.02s</strong></td>
<td>69.15 ± 0.32s</td>
<td><strong>95.89 ± 2.35%</strong></td>
</tr>
<tr>
<td>$\ell_1(d = 300)$</td>
<td>1.01 ± 0.01s</td>
<td>5519.01 ± 23.70s</td>
<td>91.20 ± 0.77s</td>
<td>93.12 ± 2.94%</td>
</tr>
<tr>
<td>$\ell_1(d = 200)$</td>
<td>0.68 ± 0.01s</td>
<td>2893.47 ± 67.41s</td>
<td>102.16 ± 1.79s</td>
<td>91.54 ± 3.15%</td>
</tr>
<tr>
<td>$\ell_1(d = 100)$</td>
<td><strong>0.35 ± 0.00s</strong></td>
<td>1068.20 ± 25.94s</td>
<td>102.13 ± 1.50s</td>
<td>86.13 ± 3.87%</td>
</tr>
<tr>
<td>Random OMP ($d = 300$)</td>
<td>1.98 ± 0.01s</td>
<td>1177.52 ± 3.02s</td>
<td>91.90 ± 0.28s</td>
<td>84.85 ± 3.43%</td>
</tr>
<tr>
<td>Random OMP ($d = 200$)</td>
<td>1.64 ± 0.02s</td>
<td>348.88 ± 1.24s</td>
<td>85.75 ± 0.15s</td>
<td>80.52 ± 4.12%</td>
</tr>
<tr>
<td>Random OMP ($d = 100$)</td>
<td>1.31 ± 0.01s</td>
<td>44.85 ± 0.78s</td>
<td>60.95 ± 0.12s</td>
<td>64.68 ± 5.50%</td>
</tr>
<tr>
<td>Hash OMP ($d = 300$)</td>
<td>4.51 ± 0.04s</td>
<td>153.08 ± 7.39s</td>
<td>63.90 ± 0.94s</td>
<td>86.92 ± 3.44%</td>
</tr>
<tr>
<td>Hash OMP ($d = 200$)</td>
<td>4.21 ± 0.02s</td>
<td>38.37 ± 2.11s</td>
<td>59.90 ± 0.39s</td>
<td>82.99 ± 3.63%</td>
</tr>
<tr>
<td>Hash OMP ($d = 100$)</td>
<td>3.93 ± 0.01s</td>
<td>7.05 ± 0.20s</td>
<td>58.33 ± 0.11s</td>
<td>64.49 ± 5.27%</td>
</tr>
<tr>
<td>Nearest Subspace</td>
<td>1.06 ± 0.06s</td>
<td>3.07 ± 0.03s</td>
<td><strong>0.07 ± 0.01s</strong></td>
<td>92.32 ± 4.16%</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Offline</th>
<th>Est</th>
<th>CheckID</th>
<th>RecRate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_2$</td>
<td>29.02 ± 0.25s</td>
<td><strong>3.55 ± 0.09s</strong></td>
<td>70.60 ± 0.71s</td>
<td><strong>98.91 ± 1.37%</strong></td>
</tr>
<tr>
<td>$\ell_1(d = 300)$</td>
<td><strong>1.52 ± 0.01s</strong></td>
<td>4191.34 ± 14.16s</td>
<td>79.48 ± 0.03s</td>
<td>96.63 ± 3.03%</td>
</tr>
<tr>
<td>Random OMP ($d = 200$)</td>
<td>2.43 ± 0.07s</td>
<td>12291.77 ± 87.31s</td>
<td>48.21 ± 0.19s</td>
<td>93.75 ± 4.40%</td>
</tr>
<tr>
<td>Hash OMP ($d = 300$)</td>
<td>7.04 ± 0.09s</td>
<td>3246.28 ± 250.37s</td>
<td>51.09 ± 0.98s</td>
<td>94.92 ± 3.86%</td>
</tr>
<tr>
<td>Nearest Subspace</td>
<td>2.74 ± 0.03s</td>
<td>3.83 ± 0.04s</td>
<td><strong>0.02 ± 0.00s</strong></td>
<td>96.87 ± 2.12%</td>
</tr>
</tbody>
</table>

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

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 $I$, then seek $\alpha$ and $\beta$ by the following $\ell_1$ minimisation:

$$\min_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^m} ||\begin{bmatrix} \alpha \\ \beta \end{bmatrix}||_{\ell_1}$$

subject to $\Phi x = \Phi [A, I][\begin{bmatrix} \alpha \\ \beta \end{bmatrix}]$, (14b) or alternatively

$$\min_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^m} ||\begin{bmatrix} \alpha \\ \beta \end{bmatrix}||_{\ell_1}$$

subject to $\Phi x = \Phi [A, I][\begin{bmatrix} \alpha \\ \beta \end{bmatrix}]$, (14b)

Identity is again determined by identifying the minimal residuals among all subjects. This is problematic, however, since $I$ can represent any possible face image without $A$. Alternatively, they construct more sophisticated features (e.g. partition features) to improve the performance of the $\ell_1$ 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 $\ell_1$ 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 $\beta$ only represent non-face objects and let $\alpha$ only represent faces. Clearly an identity $I$ 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 shown in Figure 5. The face

---

4In fact, users can design other images as long as the images follow the “key idea” mentioned above.
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 \( \hat{\mathbf{\alpha}} = [\alpha \beta] \), and then input \( \hat{\mathbf{A}} \) (instead of \( \mathbf{A} \)) to Algorithm 1 to estimate \( \hat{\mathbf{\alpha}} \) (instead of \( \mathbf{\alpha} \)). The person id is predicted via minimal residuals over all \( \mathbf{\alpha}^k \), while \( \beta \) 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 \( \ell_2 \) norm method outperforms its competitors by a very large margin in terms of recognition rate and estimation running time. In particular, \( \ell_2 \) achieves 38.33\% higher recognition rate than \( \ell_1 \) in sunglasses case and 24.33\% higher recognition rate than \( \ell_1 \) in scarves case with over 2,000 times speed up.

It is interesting to note that the image reconstructed by the \( \ell_1 \) coefficients is highly distorted (Figure 3(m)) whereas that reconstructed by \( \ell_2 \) (Figure 3(e)) is more faithful to the original image. The \( \ell_1 \) norm gives a sparse \( \mathbf{\alpha} \) whereas \( \ell_2 \) norm gives a dense one as expected (see Figure 3(k) and 3(e)). However, a sparse \( \mathbf{\alpha} \) does not necessarily lead to a more robust estimation. In fact, from Table 3, the dense \( \mathbf{\alpha} \) via the \( \ell_2 \) norm outperforms the sparse one via \( \ell_1 \) 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.

**Performance comparison against Wright et al. [18]** In [18] the downsampling of AR face images from \( 165 \times 120 \) to \( 27 \times 20 \) and \( 13 \times 10 \) is justified as being necessary in order to deal with the size of the database. 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 \( 27 \times 20 \) and \( 13 \times 10 \) as well, though the downsampling step itself is arguable: after downsampling, the \( 13 \times 10 \) 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 not only does solving (4) lead to worse performance, it is also less robust as well as 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 are also of the opinion that if $\ell_1$ 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.

8. Discussion

In this work we have compared Compressive Sensing face recognition methods, such as [18] and [14], with standard $\ell_2$ approaches. We have carried out several experiments to determine the performance of a number of different methods for robust face recognition.

The point that we want to make in this work 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 as well as 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 are also of the opinion that if $\ell_1$ 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


