2013 IEEE Conference on Computer Vision and Pattern Recognition

Online Robust Dictionary Learning

Cewu Lu Jianping Shi Jiaya Jia
The Chinese University of Hong Kong
{cwlu,jpshi,leojia}@cse.cuhk.edu.hk

Abstract

Online dictionary learning is particularly useful for processing large-scale and dynamic data in computer vision. It, however, faces the major difficulty to incorporate robust functions, rather than the square data fitting term, to handle outliers in training data. In this paper, we propose a new online framework enabling the use of ℓ^1 sparse data fitting term in robust dictionary learning, notably enhancing the usability and practicality of this important technique. Extensive experiments have been carried out to validate our new framework.

1. Introduction

Many signals and natural images can be represented as a sparse linear combination of basis vectors over an over-complete dictionary. Sparse coding and dictionary learning have been widely employed in many computer vision tasks, including image denoising [2], texture synthesis [10], and image inpainting [8], as well as higher-level tasks such as image classification [17] and unusual event detection [18].

A general framework for sparse dictionary learning is to minimize the objective function

$$\mathcal{Q}(\mathcal{B}^n, \mathbf{D}) = \frac{1}{n}\sum_{i=1}^{n}\{f(\mathbf{D}\boldsymbol{\beta}_i - \mathbf{x}_i) + \mathcal{P}(\lambda, \boldsymbol{\beta}_i)\}, \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^p$ is the i^{th} sample in set $\mathcal{X}^n = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and $\mathcal{B}^n = \{\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_n\}$. In sparse representation, each $\mathbf{x}_i$ is a sparse linear combination over a set of basis vectors from an over-complete dictionary $\mathbf{D}$ in $\mathbb{R}^{p\times q}$ with its corresponding sparse coefficient $\boldsymbol{\beta}_i$. $f(\mathbf{D}\boldsymbol{\beta}_i - \mathbf{x}_i)$ is the data fitting term and $\mathcal{P}(\cdot)$ is a sparsity penalty.

The traditional way to solve Eq. (1) involves steps to update $\mathbf{D}$ and $\mathcal{B}^n$ iteratively until convergence. Recent online strategies [8, 14] only update one $\boldsymbol{\beta}_i$ for $\mathbf{x}_i$ each time in $\mathcal{B}^n$. It is shown in [8] that this type of online solvers can achieve comparable performance as the batch processing scheme. It is because, in the training process, $\mathbf{D}$ can be considered as a combination of statistical parameters for $\mathcal{X}^n$, which does not require complete history information in $\mathbf{D}$ updating. We denote Θ_b as the batch dictionary estimator and Θ_o as the online dictionary estimator. The two types of dictionary learning are respectively expressed as

$$\mathbf{D}^n_{\text{batch}} = \Theta_b(\mathbf{x}_n, \mathcal{X}^{n-1}), \tag{2}$$
$$\mathbf{D}^n_{\text{online}} = \Theta_o(\mathbf{x}_n, \mathbf{D}^{n-1}). \tag{3}$$

In the online process, there is no need to record and process $\mathcal{X}^n$ in each round. It combines current $\mathbf{x}_n$ with $\mathbf{D}^{n-1}$ that is produced in previous iteration, statistically representable for $\mathcal{X}^{n-1}$. Thus, linearly increased computation and constant memory consumption are resulted in as data size grows. Online dictionary learning [8, 14, 13] is vastly useful for applications that involve large-scale and dynamic data, such as video sequences [11], for which traditional batch process is not suitable.

All existing online dictionary learning methods do not use the robust function in the data fitting term and might be vulnerable to large outliers. In robust statistics, ℓ^1 fitting functions were found useful to make estimation reliable. This scheme has also been extensively adopted in solving many computer vision problems. For example, in [15], robust face recognition with ℓ^1 data fitting can be accomplished even when the input faces are partly occluded. A probabilistic robust matrix factorization method was used in [16] to model background. Zhao *et al.* [19] applied robust dictionary learning to background subtraction.

Online updating for objective functions with the ℓ^1 fitting error, unfortunately, is inherently challenging. To illustrate it, we use the simplest example with only the ℓ^1-norm operator expressed as

$$\min_{\mu}\sum_{i=1}^{n}|x_i - \mu|. \tag{4}$$

The optimal μ is actually the median of $\{x_1, ..., x_n\}$. This apparently simple form, unfortunately, causes tremendous trouble for online processing because μ^n cannot be quickly estimated given only the history result $\mu^{n-1} = \text{median}(x_1, ..., x_{n-1})$ and incoming datum x_n. There is no direct relation between μ^{n-1} and μ^n. So even with the computed μ^{n-1}, finding μ^n may still require long computation time, losing all benefits brought by the online scheme. This difficulty, in fact, stems from dissimilar differentiability properties of the ℓ^1- and ℓ^2-norms. In the ℓ^2-norm optimization expressed as

$$\min_{\mu} \sum_{i=1}^{n} (x_i - \mu)^2, \tag{5}$$

online parameter estimator can be easily written as $\mu^n = \frac{n-1}{n}\mu^{n-1} + \frac{1}{n}x_n$. The problem tractability is grounded on the fact that the derivative of the ℓ^2-norm expression is a linear system with homogeneity, additivity, and superposition. In contrast, the ℓ^1 data fitting term is non-differentiable and does not have a closed-form solution.

1.1. Our Contribution

We propose an online robust dictionary learning (ORDL) algorithm. It addresses the aforementioned updating issue by converting the problem to that of minimizing several quadric functions iteratively, in which energies can be expressed in a simpler manner and their derivatives form a linear system. Our ORDL approach takes the full advantage of an online scheme – that is, it consumes constant memory as data grow and takes time linear to the size of data. In the meantime, it is very robust against outliers thanks to the sparsity nature. Our method makes dictionary learning more practical and usable in many computer vision applications, exemplified in this paper.

2. Our Approach

For a training set consisting of n elements $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i$ is normalized as zero mean and unit variance, we assume that dictionary $\mathbf{D}$ belongs to a closed, convex, and bounded set $\mathcal{D}$ as

$$\mathcal{D} \triangleq \{\mathbf{D} \in \mathbb{R}^{p\times q} \quad \text{s.t. } \forall j = 1, \ldots, q, \quad \mathbf{d}_j^T\mathbf{d}_j \le 1\}, \tag{6}$$

where $\mathbf{d}_j$ is the j^{th} column of $\mathbf{D}$. Previous online dictionary learning [8, 14] is based on the function

$$\min_{\mathbf{D}\in\mathcal{D},\mathcal{B}} \frac{1}{n}\sum_{i=1}^{n} \left\{\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\beta}_i\|_2^2 + \lambda\|\boldsymbol{\beta}_i\|_1\right\}, \tag{7}$$

where $\mathcal{B} = \{\boldsymbol{\beta}_1, , ..., \boldsymbol{\beta}_n\}$, and λ is a regularization parameter. Different from it, our robust dictionary learning is defined as

$$\min_{\mathbf{D}\in\mathcal{D},\mathcal{B}} \frac{1}{n}\sum_{i=1}^{n} \{\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\beta}_i\|_1 + \lambda\|\boldsymbol{\beta}_i\|_1\}. \tag{8}$$

As described in Section 1 and in [15, 19], $\|\mathbf{x}_i - \mathbf{D}\boldsymbol{\beta}_i\|_1$ makes date fitting less vulnerable to outliers. We will present several applications later that can be profitted from this model.

Algorithm 1 Online Robust Dictionary Learning

```
1: Input: signals $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, regularization parameter $\lambda$, initial dictionary $\mathbf{D}_0$, mini-batch data size $h$
2: for each $\{\mathbf{x}_{t-h+1}, \ldots, \mathbf{x}_t\}$ in $\mathcal{X}$ do
3:     for $j = t-h+1 : t$ do
4:         update $\hat{\boldsymbol{\beta}}_j = \arg\min_{\boldsymbol{\beta}_j} \|\mathbf{x}_j - \mathbf{D}\boldsymbol{\beta}_j\|_1 + \lambda\|\boldsymbol{\beta}_j\|_1$
5:     end for
6:     update $\mathbf{D}_t$ by combining $\mathbf{D}_{t-h}$ and $\{\hat{\boldsymbol{\beta}}_{t-h+1}, \ldots, \hat{\boldsymbol{\beta}}_t\}$.
7:     $t = t + 1$.
8: end for
9: Return $\mathbf{D}_n$.
```

By tradition, online dictionary learning updates $\mathbf{D}$ and robust sparse coding estimates $\mathcal{B}$, which is achieved by iterative reweighed least squares discussed in [19]. Algorithm 1 lists the steps.

In what follows, matrices are denoted by capital bold letters. The i^{th} row of $\mathbf{A}$ is $\mathbf{A}(i, :)$ and the i^{th} column of $\mathbf{A}$ is $\mathbf{A}(:, i)$. The entry of $\mathbf{A}$ in the i^{th} row and j^{th} column is a_{ij}.

2.1. Batch Dictionary Update

We begin with the batch dictionary update procedure and then extend it to the online version. Theoretical robustness analysis of our online dictionary update is provided.

In batch robust dictionary learning [19], updating of the dictionary assumes that all elements in $\mathcal{B}$ are provided. We denote matrix $\mathbf{X}$ as $[\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{p\times n}$ and $\mathbf{B}$ as $[\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_n] \in \mathbb{R}^{q\times n}$ in order to distinguish them from those in the online procedure. Batch dictionary updating is accordingly expressed as

$$\min_{\mathbf{D}} \frac{1}{n}\|\mathbf{X} - \mathbf{D}\mathbf{B}\|_1, \tag{9}$$

which is a standard ℓ^1-regression problem. It cannot be solved the same way as ℓ^2-regression owing to the lack of differentiability. We resort to the iterative reweighted least squares (IRLS) [3].

Since each $\mathbf{D}(j, :)$ $(j = 1, ..., p)$ can be estimated independently, without loss of generality, we express the j^{th} function optimization as

$$\mathbf{D}(j, :) = \arg\min_{\mathbf{d}} \frac{1}{n}\sum_{i=1}^{n} |x_{ij} - \mathbf{d}\boldsymbol{\beta}_i|, \tag{10}$$

where $\mathbf{d} \in \mathbb{R}^{1\times q}$ and x_{ij} is the j^{th} element of $\mathbf{x}_i$. The IRLS scheme solves the following two problem in each iteration

Algorithm 2 Batch Dictionary Update

```
1: input: $\mathbf{x}_1, \cdots, \mathbf{x}_n$; $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_n$; random matrix $\mathbf{D}$;
   $w_i^j = 1$ for all $i = 1, \cdots, n, j = 1, \cdots, p$
2: repeat
3:   for $j = 1 : p$ do
4:     $\mathbf{M}^j = \sum_{i=1}^n w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T$
5:     $\mathbf{C}^j = \sum_{i=1}^n w_i^j x_{ij} \boldsymbol{\beta}_i^T$
6:     solve linear system $\mathbf{C}^j = \mathbf{D}(j,:)\mathbf{M}^j$
7:     calculate $w_i^j$ $(i = 1, 2, ..., n)$ by Eq. (12)
8:   end for
9: until converges
10: output: $\mathbf{D}$
```

until convergence:

$$\mathbf{D}(j,:) = \arg\min_{\mathbf{d}} \frac{1}{n} \sum_{i=1}^n w_i^j (x_{ij} - \mathbf{d}\boldsymbol{\beta}_i)^2. \qquad (11)$$

w_i^j is computed as

$$w_i^j = \frac{1}{\sqrt{(x_{ij} - \mathbf{D}(j,:)\boldsymbol{\beta}_i)^2 + \delta}}, \qquad (12)$$

where δ is a small positive value ($\delta = 0.0001$ in our experiments). Each iteration of Eq. (11) involves minimizing a quadratic objective function. The global optimum can be reached by taking derivatives and setting them to zeros. This leads to solving $\mathbf{D}(j,:)$ in the linear system

$$[\sum_{i=1}^n w_i^j x_{ij} \boldsymbol{\beta}_i^T] = \mathbf{D}(j,:)[\sum_{k=1}^n w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T]. \qquad (13)$$

The detailed steps are provided in Algorithm 2 where coefficients $\mathbf{M}^j = \sum_{i=1}^n w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T$ and observation $\mathbf{C}^j = \sum_{i=1}^n w_i^j x_{ij} \boldsymbol{\beta}_i^T$, corresponding to the terms in Eq. (13).

2.2. Online Dictionary Update

Extending the above process to online is not straightforward because we only see new mini-batch data $\mathbf{x}_{t-h+1}, ..., \mathbf{x}_t$, along with statistical records of history data $\mathbf{x}_1, ..., \mathbf{x}_{t-h}$ and their coefficients $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_{t-h}$. The statistical records are denoted as $\mathbf{M}_{t-h}^j$ and $\mathbf{C}_{t-h}^j$ similar to those in Algorithm 2. Their update is expressed as

$$\mathbf{M}_t^j = \mathbf{M}_{t-h}^j + \sum_{i=t-h+1}^t w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T, \mathbf{C}_t^j = \mathbf{C}_{t-h}^j + \sum_{i=t-h+1}^t w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T,$$

where $\mathbf{M}_{t-h}^j = \sum_{i=1}^{t-h} w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T$ and $\mathbf{C}_{t-h}^j = \sum_{i=1}^{t-h} w_i^j x_{ij} \boldsymbol{\beta}_i^T$. Now, information for new data is stored in $\sum_{i=t-h+1}^t w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T$ and $\sum_{i=t-h+1}^t w_i^j x_{ij} \boldsymbol{\beta}_i^T$.

With this configuration, we update $\mathbf{D}_t$ using Algorithm 3 in an online style. To solve the linear system in line 6 of Algorithm 3, we employ a conjugate gradient method, taking previous $\mathbf{D}_{t-h}$ as initialization in the current round. Because matrix $\mathbf{M}_t^j$ is often diagonally dominated, reasonable initialization from existing data can make conjugate gradient updating converge quickly. We do not want any column of $\mathbf{D}$ to be arbitrarily large, and thus optionally introduce constrained optimization

$$\mathbf{D}(:,i) \leftarrow \frac{\mathbf{D}(:,i)}{\max\{1, \|\mathbf{D}(:,i)\|_2^2\}} \qquad (14)$$

in each round. In practice, thanks to the warm start of $\mathbf{D}_{t-h}$, dictionary update in Algorithm 3 converges in $5 \sim 7$ iterations.

Algorithm 3 Online Robust Dictionary Update

```
1: input: new mini-batch data $\mathbf{x}_{t-h+1}, ..., \mathbf{x}_t$, coefficients
   $\boldsymbol{\beta}_{t-h+1}, ..., \boldsymbol{\beta}_t$; $\mathbf{M}_{t-h}^j$, $\mathbf{C}_{t-h}^j$ for all $j$; $\mathbf{D} = \mathbf{D}_{t-h}$ and
   $w_i^j = 1$ for all $i = t-h+1, ..., t, j = 1, ..., p$
2: repeat
3:   for $j = 1 : p$ do
4:     $\mathbf{M}_t^j \leftarrow \mathbf{M}_{t-h}^j + \sum_{i=t-h+1}^t w_i^j \boldsymbol{\beta}_i \boldsymbol{\beta}_i^T$
5:     $\mathbf{C}_t^j \leftarrow \mathbf{C}_{t-h}^j + \sum_{k=t-h+1}^t w_i^j x_{ij} \boldsymbol{\beta}_i^T$
6:     solve linear system $\mathbf{C}_t^j = \mathbf{D}(j,:)\mathbf{M}_t^j$
7:     calculate $w_i^j$ $(i = t-h+1, ..., t)$ by Eq. (12)
8:   end for
9: until converge
10: output: $\mathbf{D}_t = \mathbf{D}$, $\mathbf{M}_t^j$, $\mathbf{C}_t^j$
```

Algorithm 4 IRLS solver for Eq. (15)

```
1: input: $\mathbf{x}_{t-h+1}, ..., \mathbf{x}_t$, $\boldsymbol{\beta}_{t-h+1}, ..., \boldsymbol{\beta}_t$, $\overline{w}_i^j$ $(i =
   1, \ldots, t-h)$, all $w_i^j = 1$
2: repeat
3:   calculate $\mathbf{d}$ by Eq. (15)
4:   calculate $w_i^j$ $(i = t-h+1, ..., t)$ by Eq. (12)
5: until converge
6: output: $\mathbf{D}_t = \mathbf{D}$, $\mathbf{M}_t^j$, $\mathbf{C}_t^j$
```

2.3. Analysis

We analyze how robust the online algorithm is. We demonstrate that updating the j^{th} row of dictionary in Algorithm 3 is equivalent to solving

$$\min_{\mathbf{d}} \sum_{i=1}^{t-h} \overline{w}_i^j (x_{ij} - \mathbf{d}\boldsymbol{\beta}_i)^2 + \sum_{i=t-h+1}^t |x_{ij} - \mathbf{d}\boldsymbol{\beta}_i|, \qquad (15)$$

where $\overline{w}_i^j$ $(i = 1, \ldots, t-h)$ is the converged history weight calculated by Eq. (12). According to [3], we can solve Eq. (15) using IRLS as Algorithm 4.

Obviously, line 3 in Algorithm 4 is equivalent to lines 4-6 in Algorithm 3. Therefore, estimation of each row of the dictionary in Algorithm 3 is achieved by solving Eq. (15).

The robustness property to resist outliers in Algorithm 3 is preserved in Eq. (15).

We analyze effectiveness of Eq. (15) in what follows.

1. At the beginning of training, term $\sum |x_{ji} - \mathbf{d}\boldsymbol{\beta}_i|$ plays an important role in outlier rejection. According to robust statistics [12], high quality results can be obtained if the number of outliers in the coming data is smaller than $h/2$. An outlier sample x_{ij} will find a small corresponding weight $\overline{w}_i^j$ according to Eq. (12).

2. As training data grow, the history term carries inlier information, in turn enhancing outlier rejection by Eq. (15) and even helping handle difficult problems where outliers in the new mini-batch data are less than $h/2$.

We note although this scheme is robust, it still cannot handle the extreme case where the initial data are primarily outliers and following ones are all inliers. But, this case is rare in real-world applications and can be avoided generally.

2.4. Connection between Online and Batch Schemes

The online method seemingly differs from the batch version. But, in fact, the difference in terms of robustness is rather small. To understand it, we rewrite the batch update as

$$\min_{\mathbf{d}} \sum_{i=1}^{t-h} |x_{ij} - \mathbf{d}\boldsymbol{\beta}_i| + \sum_{i=t-h+1}^{t} |x_{ij} - \mathbf{d}\boldsymbol{\beta}_i|. \tag{16}$$

Compared to Eq. (15), the difference is on the history information part. Online robust dictionary learning (ORDL) has a weighted square, i.e., $\sum \overline{w}_i^j (x_{ij} - \mathbf{d}\boldsymbol{\beta}_i)^2$, where outliers get small weights. In comparison, batch robust dictionary learning (BRDL) contains the ℓ^1-norm term $\sum |x_{ij} - \mathbf{d}\boldsymbol{\beta}_i|$. In robust statistics [12], this weighted square is a reasonable approximation of the ℓ^1-norm. Practically, the two approaches perform similarly, as verified in our experiments.

3. Experiments

We qualitatively and quantitatively evaluate our online framework on different sets of data and applications. There has been research [19] showing that robust dictionary learning incorporating the ℓ^1 data term outperforms that using the ℓ^2 data constraint when outliers exist. We show more evidences, especially using the online scheme. All the algorithms are implemented using MATLAB on a PC with 2.60GHz CPU and 2.0GB memory.

3.1. Evaluation on Synthetic Data

The performance of online robust dictionary update is compared with that of the batch process. We generate n sparse coefficients $[\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_n] \in \mathbb{R}^{100\times n}$ with 20% sparsity ratio, where n is the number of training data up to 16000. Given a dictionary $\mathbf{D} \in \mathbb{R}^{32\times 100}$, we generate training data $\mathbf{x}_i = \mathbf{D}\boldsymbol{\beta}_i + \xi_i (1 \le i \le n)$, where ξ_i is additive Laplace noise to simulate outliers. The size of incoming (mini-batch) data is $h = 50$. Experiments are carried out under different noise levels and with different sizes of data.

We measure the difference between the batch dictionary result $\mathbf{D}_b$ and the online one $\mathbf{D}_o$ via correlation $Corr(\mathbf{D}_b, \mathbf{D}_o)$. We set $\lambda = 0.15$. Fig. 1(a) shows when the training data size t is large, $\mathbf{D}_o$ approaches $\mathbf{D}_b$ in all noise levels.

We also evaluate how our online scheme updates the resulting dictionary by measuring the dictionary difference $\|\mathbf{D}_o^t - \mathbf{D}_o^{t-1}\|_2^2$ with regard to the input data size t. Fig. 1(b) shows this cost quickly and monotonically decreases when data grow. This proves that our method converges practically and the optimization is robust against outlier.

We also compare running time and memory consumption and plot them in Fig. 2. We conclude that our online dictionary update uses linear time and nearly constant memory. The batch process, on the contrary, takes polynomial

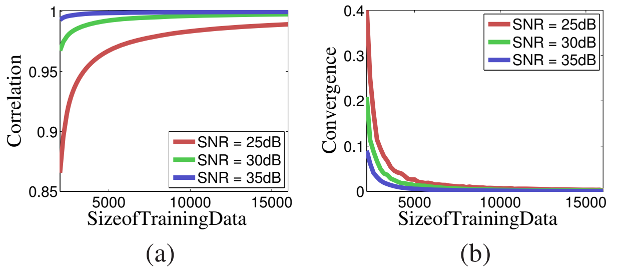


Figure 1. (a) Correlation between $\mathbf{D}_b$ and $\mathbf{D}_o$ v.s. training data size t. (b) $\|\mathbf{D}_o^t - \mathbf{D}_o^{t-1}\|_2^2$ v.s. data size t. Noise levels are 25 dB, 30 dB, and 35 dB.

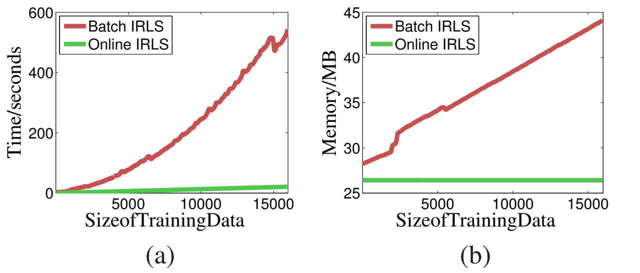


Figure 2. (a) Running time comparison. (b) Memory consumption comparison.

SNR	BRDL	ORDL	KSVD [2]	Mairal *et al.* [8]
15dB	0.051	0.067	0.412	0.505
20dB	0.034	0.042	0.224	0.282
25dB	0.021	0.030	0.121	0.159

Table 1. Reconstruction error under different outlier levels.

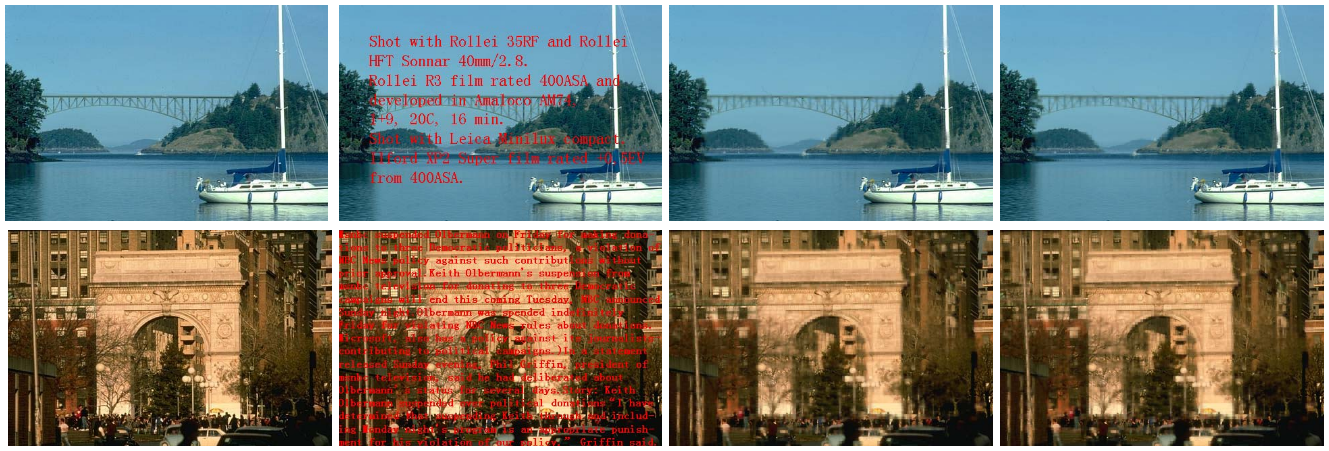


Figure 3. Image inpainting results by online and batch dictionary learning. (a) Original images. (b) Damaged images. (c) Restored images by our method. (d) Restored images in the batch scheme.

time and memory linear to the data size.

We carry out another synthetic experiment to evaluate the robustness of our online method. We use more data with $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{64\times16000}$. The data generation procedure is similar to the previous one. We train a dictionary with $\mathbf{D} \in \mathbb{R}^{64\times512}$. The robustness is measured via total reconstruction error $\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\beta_i\|_2^2$. If a learning method can reject outlier in the training data, the total reconstruction error should be small. We compare different methods under various outlier levels in Table 1. It shows that the performance of our online robust dictionary learning (ORDL) is comparable to batch robust dictionary learning (BRDL) and outperforms those using the ℓ^2 data term significantly.

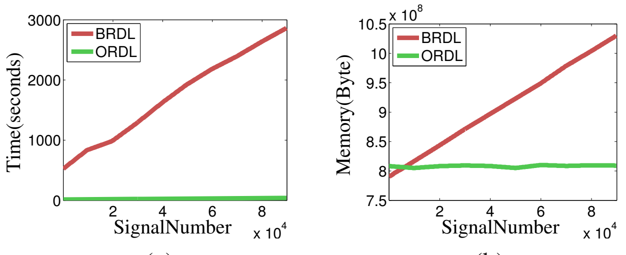
3.2. Evaluation on Natural Image Inpainting

We evaluate our online robust dictionary learning using color image inpainting, which aims to infer missing pixels due to superimposed text, subtitles, or watermark. In this application, the dictionary is trained on natural image datasets [9] with 15% random outliers. Laplacian noise with scale 30 is used here. We apply the same sparse coding method with the parameters set the same as those in the previous case. In the reconstruction step, to infer missing pixels, we minimize

$$\frac{1}{2}\|\mathbf{M} \bullet (x_i - \mathbf{D}\boldsymbol{\beta}_i)\|_1 + \lambda\|\boldsymbol{\beta}_i\|_1,$$

where $\mathbf{M}$ is a mask vector to include missing pixels. $\bullet$ denotes element-wise multiplication. Initial $\mathbf{D}$ is generated as standard Gaussians, and is then normalized for each column as a unit vector.

The results are shown in Fig. 3. Basically, the batch and online processes produce very similar results; but the online



(a) (b)

Figure 4. Performance comparison. (a) Running time comparison. (b) Memory consumption comparison.

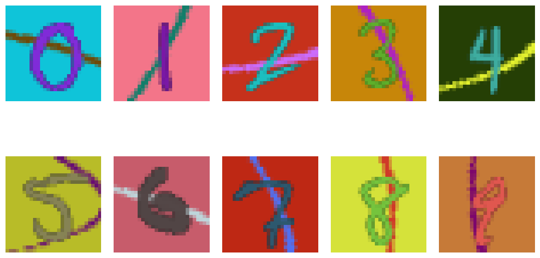


Figure 5. Example of outlier contaminated color digit images.

process significantly reduces the running time. For comparison, considering the dictionary update step only, the online procedure takes 0.7 second on average to accept a new mini-batch input with $h = 256$, and 40 seconds to complete 90000 atoms. In the batch mode, it takes about 3000 seconds in total. We report the time and memory consumption in Fig. 4 to demonstrate the advantages of our method.

3.3. Evaluation on Digit Recognition

Digit recognition is an important tool in many systems involving document processing. This task requires a large number of training data. However, these training images

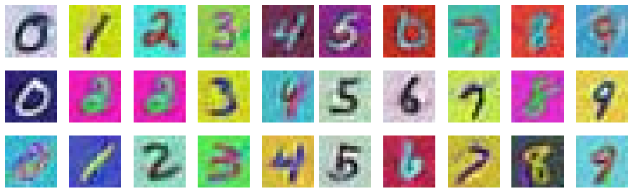


Figure 6. A few learned dictionary atoms using our Online Robust Dictionary Learning (ORDL).

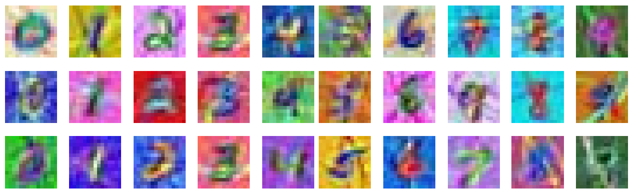


Figure 7. A few learned dictionary atoms using the ℓ^2 data-term based dictionary learning method (KSVD [2]).

could be contaminated in different ways. Robust dictionary learning remedies part of these problems.

We demonstrate, for this important application, our method can accomplish results robustly with short running time and reduced memory consumption. Our experiments are carried out on handwritten digit datasets MNIST [6] and USPS [1]. The MNIST dataset has 60000 training images and 10000 testing images. The USPS dataset has 7291 images for training and 2007 for testing. To verify the robustness of our approach, we add random outliers to the images (both training and testing data) and colorize them to greatly increase the versatility and complexity given the digits all in grayscale originally. The background and digits are assigned with different colors. The outliers are generated by a high-order polynomial curve with random parameters. Fig. 5 shows several examples.

We resize each image to 15×15 pixels. All pixels from the RGB channels are concatenated together as a final feature vector. We learn separate dictionaries for each digit via outlier contaminated training data. In the testing phase, given a digit image, we reconstruct it over the ten learned dictionaries and choose the one with smallest reconstruction error. We compare our online robust learning (ORDL) with batch robust learning (BRDL) and other state-of-the-art ℓ^2-norm data term dictionary learning methods, including KSVD [2] and Mairal *et al.* [8]. The learned dictionary by our method is shown in Fig. 6, which rejects outliers. Compared to the ℓ^2-norm data-term dictionary learning results in Fig. 7, more digit information is captured in our atoms. Table 2 lists the recognition errors. Our ORDL can achieve a comparable recognition rate with BRDL, while it outperforms a new alternatives. We also compare the running time between BRDL and ORDL, where the quantities are listed in Table 3.

Dataset	BRDL	ORDL	KSVD [2]	Mairal *et al.* [8]
MNIST	18.1	22.7	39.2	34.3
USPS	27.3	29.4	45.3	42.5

Table 2. Error rate (%) for the digit recognition task using different methods on the MNIST and USPS datasets.

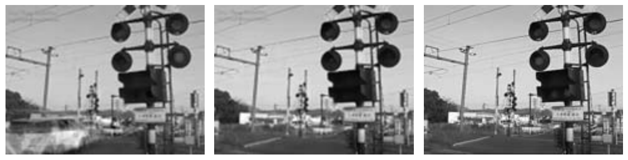

(a) (b) (c)

Figure 8. Background subtraction by (a) K-SVD [2] (b) batch robust dictionary update, and (c) our method with online dictionary update. Our results do not contain the "ghost" artifacts, similar to those produced by batch dictionary learning.

3.4. Evaluation on Background Subtraction

We further evaluate our method on background subtraction. This application takes video sequences as input. Each frame is compared against a background model. Pixels that significantly deviate from the model are regarded as foreground. In [5, 4], it is cast as a sparse signal recovery problem. That is, given the i^{th} frame $\mathbf{x}_i$, its background is assumed to be sparsely represented as a linear combination of a few atoms in the learned dictionary $\mathbf{D}$. Suppose $\boldsymbol{\beta}_i$ is the i-th coefficient. Foreground is represented as $\mathbf{x}_i - \mathbf{D}\boldsymbol{\beta}_i$, which is assumed to be sparse after background subtraction.

In our model, background subtraction can be achieved by solving our robust dictionary learning model

$$\min_{\boldsymbol{\beta}_i} \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\beta}_i\|_1 + \lambda\|\boldsymbol{\beta}_i\|_1. \tag{17}$$

Given a learned $\mathbf{D}$, the background can be reconstructed using $\mathbf{D}\boldsymbol{\beta}_i^*$, where $\boldsymbol{\beta}_i^*$ is the optimal solution of Eq. (17). In this experiment, we use the background substraction dataset of [5]. The video resolution is 120×160. Each frame is converted into a single-channel image using [7] to reduce computation.

It is notable that $\mathbf{D}$ is trained using all existing frames in previous work [19]. However, a traffic video contains vehicles and pedestrians, which are significantly different from the background regions. These outliers may result in an inappropriate dictionary. For the example shown in Fig. 8, when the dictionary is learned with the ℓ^2 data fitting term, many moving vehicles (i.e., outliers) are *averaged* into background. To reject outliers, the method of [19] also introduces the ℓ^1 data term. However, it works in a batch mode, difficult for long-video-sequence processing. Our framework not only saves time and memory, but also can help produce high quality foreground segments.

Digit	0	1	2	3	4	5	6	7	8	9
Data Size	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
ORDL	187	214	188	194	185	171	187	198	185	188
BRDL	2424	3466	2440	2587	2392	1979	2359	4115	2344	2402

Table 3. Time comparison (in *second*) in digit recognition in the MNIST dataset for each digit.

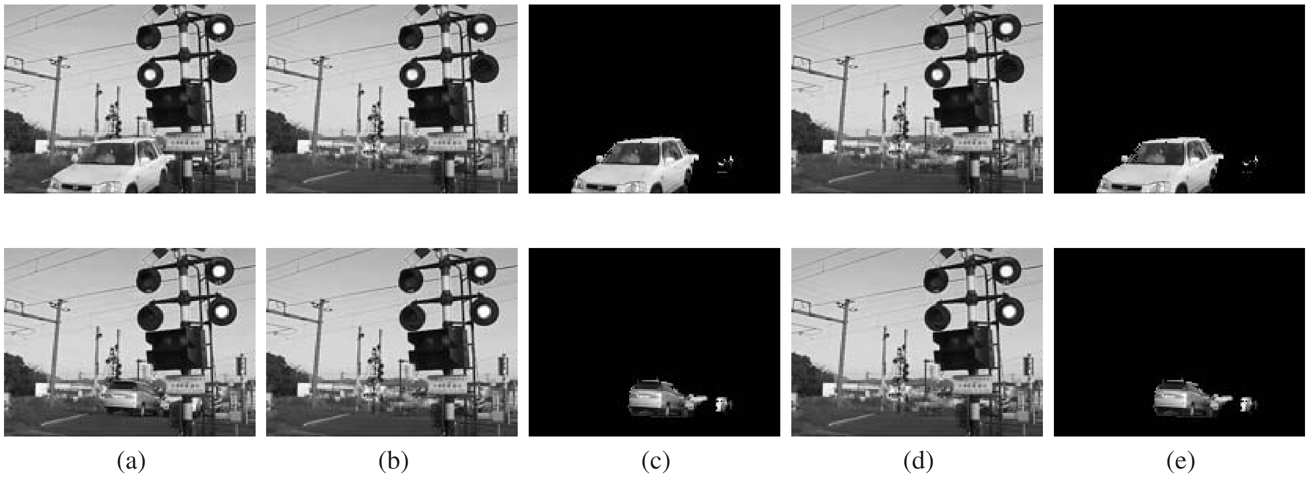


Figure 9. Two traffic light statuses. (a) input frames; (b) subtracted background with batch dictionary update; (c) foreground in batch mode; (d) background with online dictionary update; (e) foreground by our online algorithm.

In our online dictionary update, new data (or min-bacth) are with size $h = 20$. We compare batch and online results with those of K-SVD [2] in Figs. 8 and 9. Our results do not contain the "ghost" artifacts cause by outliers. Our online dictionary learning takes much less time than the batch scheme, as compared in Fig. 10.

In Fig. 11, we compare the extracted foreground segments by our online dictionary learning method, the batch one [19], and by the method using the ℓ^2-norm configuration. Following [19], in each sequence, we have 10-frame ground truth foreground. Based on them, we quantitatively compare the precision, recall, and F-score. Results are listed in Table 4. it shows that our online robust dictionary learning method can achieve comparable accuracy with batch robust dictionary learning and outperforms those ℓ^2 data term based dictionary methods. Online dictionary update uses 1.1 seconds on a PC.

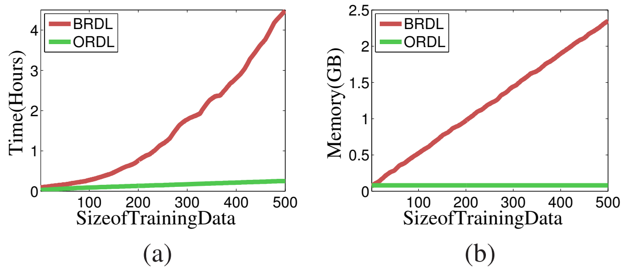


Figure 10. Comparison of time and memory. (a) Running time. (b) Memory Consumption. BRDL: batch robust dictionary learning; ORDL: online robust dictionary learning.

4. Conclusion

We have proposed an online robust dictionary learning framework. The online dictionary update scheme saves time and memory while not sacrificing much accuracy. We have developed an algorithm with full details to accomplish it and have shown that this algorithm can produce similar-quality results as the batch robust one. We believe this framework gives an unconventional way to solve many practical problems containing dynamic and large-size data and will vastly and widely benefit computer vision systems.

Acknowledgements

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region (Project No. 412911) and by NSF of China (key project No. 61133009).

References

[1] Usps handwritten digits dataset. www-i6.informatik.rwth-aachen.de/.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse

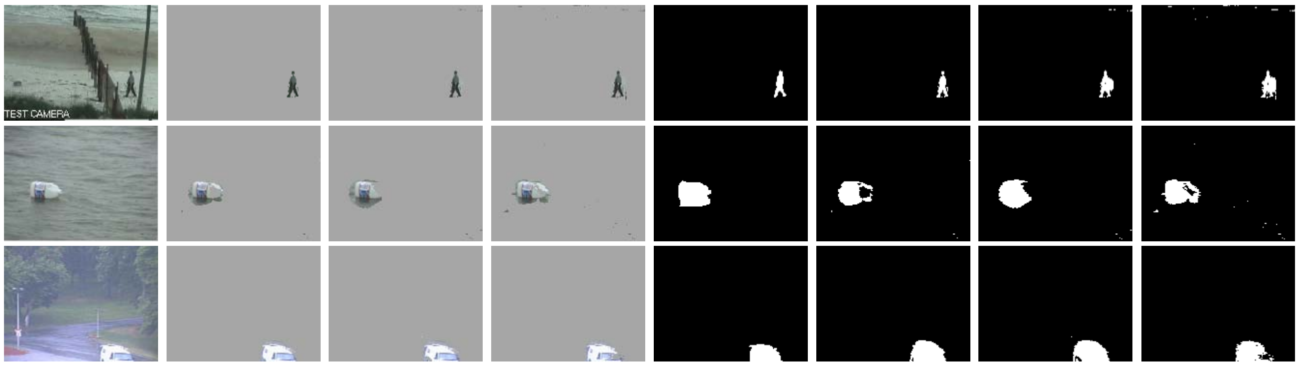


Figure 11. Comparison of background subtraction. (a) inputs; (b) our results using online dictionary update; (c) results of [19] using an ℓ^1-norm data fitting term; (d) sparse coding and dictionary update with the ℓ^2-norm data fitting term; (e) ground truth foreground masks; (f)-(h) foreground masks corresponding to (b)-(d).

Videos	Ocean man			Water Object			Rain Car		
	Pre	Rec	F-score	Pre	Rec	F-score	Pre	Rec	F-score
ORDL	0.8141	0.8148	**0.7922**	0.6752	0. 7548	**0.7128**	0.9236	0.7943	0.8541
BRDL	0.7552	0.7982	0.7761	0.7009	0.7225	0.7115	0.9489	0.7920	**0.8634**
Mairal *et al.* [8]	0.6367	0.8518	0.7287	0.6312	0.6149	0.6229	0.7940	0.8010	0.7975

Table 4. Precision, recall and F-score for the example shown in Fig. 11 using different dictionary methods.

representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[3] N. Bissantz, L. Dmbgen, A. Munk, and B. Stratmann. Convergence analysis of generalized iteratively reweighted least squares algorithms on convex function spaces. *SIAM Journal on Optimization*, 19:1828–1845, 2009.

[4] V. Cevher, A. Sankaranarayanan, M. Duarte, D. Reddy, R. Baraniuk, and R. Chellappa. Compressive sensing for background subtraction. *ECCV*, pages 155–168, 2008.

[5] J. Huang, X. Huang, and D. N. Metaxas. Learning with dynamic group sparsity. In *ICCV*, pages 64–71, 2009.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[7] C. Lu, L. Xu, and J. Jia. Real-time contrast preserving decolorization. In *SIGGRAPH ASIA Posters*, page 16, 2012.

[8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[9] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, July 2001.

[10] G. Peyre. Sparse modeling of textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, 2009.

[11] M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *TIP*, 19:27–35, 2009.

[12] W. Rey. *Introduction to robust and quasi-robust statistical methods*. Springer-Verlag New York:, 1983.

[13] J. Shi, X. Ren, G. Dai, J. Wang, and Z. Zhang. A non-convex relaxation approach to sparse dictionary learning. In *CVPR*, pages 1809–1816, 2011.

[14] Z. Szabo, B. Poczos, and A. Lorincz. Online group-structured dictionary learning. In *CVPR*, pages 2865–2872, 2011.

[15] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Towards a practical face recognition system: Robust alignment and illumination by sparse representation. *PAMI*, 34(2):372–386, 2012.

[16] N. Wang, T. Yao, J. Wang, and D.-Y. Yeung. A probabilistic approach to robust matrix factorization. In *ECCV*, pages 126–139, 2012.

[17] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009.

[18] B. Zhao, L. Fei-Fei, and E. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *CVPR*, pages 3313–3320, 2011.

[19] C. Zhao, X. Wang, and W.-K. Cham. Background subtraction via robust dictionary learning. *EURASIP Journal on Image and Video Processing*, 2011.