This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Image Transformation based on Learning Dictionaries across Image Spaces

Kui Jia, Xiaogang Wang, and Xiaoou Tang, *Fellow, IEEE*

Abstract—In this paper, we propose a framework of transforming images from a source image space to a target image space, based on learning coupled dictionaries from a training set of paired images. The framework can be used for applications such as image super-resolution, and estimation of image intrinsic components (shading and albedo). It is based on a local parametric regression approach, using sparse feature representations over learned coupled dictionaries across the source and target image spaces. After coupled dictionary learning, sparse coefficient vectors of training image patch pairs are partitioned into easily retrievable local clusters. For any test image patch, we can fast index into its closest local cluster and perform a local parametric regression between the learned sparse feature spaces. Obtained sparse representation (together with the learned target space dictionary) provides multiple constraints for each pixel of the target image to be estimated. The final target image is reconstructed based on these constraints. The contributions of our proposed framework are three-fold. (1) We propose a concept of coupled dictionary learning based on coupled sparse coding, which requires the sparse coefficient vectors of a pair of corresponding source and target image patches have the same support, i.e., the same indices of nonzero elements. (2) We devise a space partitioning scheme to divide the high-dimensional but sparse feature space into local clusters. The partitioning facilitates extremely fast retrieval of closest local clusters for query patches. (3) Benefiting from sparse feature based image transformation, our method is more robust to corrupted input data, and can be considered as a simultaneous image restoration and transformation process. Experiments on intrinsic image estimation and super-resolution demonstrate the effectiveness and efficiency of our proposed method.

Index Terms—Image transformation, Image mapping, Sparse coding, Intrinsic images, Super-resolution.

1 Introduction

Many problems in computer vision can be cast as transforming images from one form to another. In this work, we assume that the pairs of images have the same content and are in registration with each other, but are visually different. Typical applications include, but not limited to, super-resolution, face mapping of different styles, estimation of *intrinsic* images, such as shading and albedo images, and various non-photorealistic rendering. Some existing methods generate stylistic images based on a single or a pair of reference images [3], [1], [2]. Others learn the mapping relations from large training sets of paired images [4], [6], [7], [8], [49], [51]. Our work belongs to the second category. A few representative methods of such kind are summarized as follows.

Chang *et al.* [7] assumed that image patches in source and target spaces form manifolds with similar local geometries, and a target patch could be generated as a linear combination of its neighbors. This approach was applied to super-resolution. Tappen *et al.* [11] proposed an ExpertBoost algorithm to learn a nonlinear Mixture of Experts estimator, which was parameterized as a set of prototype patches and their corresponding linear regression coefficients. It was applied to image denoising, and estimating intrinsic shading/albedo images. Spiritually similar, Kim and Kwon [12] adopted sparse kernel ridge regression to learn a mapping function from paired training patches, and applied it to super-resolution.

In general, transforming images from a source space to a target space is difficult. Natural images are very high dimensional, statistically non-Gaussian, and exhibit abundant varying texture patterns. Directly learning image models and their mapping relations may not be feasible. Even working on local patches, the dimensionality is still too high to learn a good and explicit mapping function. The above reviewed methods either use non-parametric approaches, such as nearest neighbor (NN) searching, and learn the mapping relations using the found neighbor training patch pairs [4], [7], or use non-linear regression based on summarizing the training data using a small number of prototype patches [11], [12]. When dealing with a huge amount of training patches, searching NNs could be prohibitively slow, and also costs large memory. Although there exist acceleration methods such as approximate nearest neighbor (ANN) search [22], [23], however, they cannot generally cope with the memory cost problem. For nonlinear regression [11], [12], when the prototype number is getting large, significant computation is required, while fewer prototypes cannot well approximate the image space.

We are motivated to find alternative solutions that can address these major challenges. In particular, we consider the more direct approach of learning parametric models. Note that for an input patch, non-parametric approaches search its NNs at runtime, and those NNs

- *K. Jia is with Advanced Digital Sciences Center, University of Illinois at Urbana-Champaign. E-mail: kuijia@gmail.com.*
- *X. Tang is with Department of Information Engineering, The Chinese University of Hong Kong. E-mail: xtang@ie.cuhk.edu.hk.*
- *X. Wang is with Department of Electronic Engineering, The Chinese University of Hong Kong. E-mail: xgwang@ee.cuhk.edu.hk.*

Digital Object Indentifier 10.1109/TPAMI.2012.95

correspond to a neighborhood of the query in the image space. If one can divide the space into neighborhoods that are precise and representative, and learn mapping relations within each neighborhood offline, it will be not necessary to store all training samples and online search NNs, without sacrifice of the mapping accuracy. In literature, vector quantization [45], such as k-means clustering, is popularly used to partition a given space into representative neighborhoods. However, in high-dimensional spaces it is very difficult to achieve precise quantization which requires a huge number of neighborhoods. It requires high computational complexity to learn and retrieve those neighborhoods, and also large memory to store the neighborhood centroids. One may consider other simpler methods such as scalar or lattice quantizers [45], however, their performance is usually poor since their assumptions of data distribution are rarely satisfied by natural images.

The difficulties of working in the original image space motivate us to consider doing the mappings in some projected feature spaces. Recently, sparse representation over a learned dictionary [26] has achieved state-of-the-art performance for many image restoration applications [25], [27] as well as various image classification tasks [34], [31], [35]. It shows that over-complete sparse models can be well adapted to natural images. Motivated by such progress, we are tempted to learn two *separate* dictionaries for both the source and target image spaces, and perform partitioning of the learned sparse feature spaces for local parametric image transformation. Instead of doing *independent* dictionary learning for source and target image spaces, we introduce the concept of *coupled dictionary learning* based on *coupled sparse coding*. The coupling is realized by enforcing sparse coefficient vectors of a pair of corresponding source and target image patches have the same support, i.e., the same indices of nonzero elements. We use an efficient *active set block coordinate gradient descent (BCGD)* algorithm for coupled sparse coding, which can also efficiently approximate the solution path [42], [41]. Coupled dictionary learning bridges the processes of source and target dictionary learning, resulting in dictionary atoms that well model and correlate different image spaces. Indeed, coupled sparse coding of image patch pairs can identify the most *significant*, *reduced*, and *correlated* feature subspaces. We devise an efficient scheme to partition the high-dimensional (*in case of over-complete dictionary*) but sparse (*for nonzero coefficients*) feature spaces of training patch samples into easily retrievable local clusters (cells). Our scheme exploits the information of both support patterns and coefficient values of sparse feature vectors, and facilitates efficient retrieval of the closest cluster for a query patch. Our aim is that by doing image mappings in the learned sparse feature spaces, both mapping accuracy and mapping efficiency can be achieved. The mapping efficiency can be realized by local parametric regression and is based on our proposed space partitioning method that facilitates efficient retrieval of local clusters. The mapping accuracy comes from two facts: (1) sparse feature vectors of training patch samples are highly close to each other within local clusters; (2) it is much easier to model the mapping relations in reduced, significant and correlated feature subspaces.

Given a query source patch, we compute its sparse feature vector, and fast index into its closest local cluster, and then estimate its sparse vector over the optimally learned target space dictionary by local linear regression. The mapping relations have been offline learned. Figure 1 gives a graphical illustration. By this way, we only need to store the space indexing structure and regression parameters within each local cluster. Our method can effectively deal with the speed and memory cost challenges mentioned above. Compared with alternative approaches that do not enforce coupled sparsity, the number of parameters to be learned and stored is much reduced, and the local regression process is also more stable (cf. Section 3.1.1).

In this paper, we apply our framework to two applications, namely estimating the intrinsic components (shading and albedo) of an observed image, and image super-resolution. For the latter one, we achieve performance comparable with the state-of-the-art. Our results on intrinsic image estimation outperform current single-input-image based methods. Our method is potentially more robust to corrupted data. In intrinsic image estimation, if noisy images are used as input, we get increased performance gap compared with other methods. In this sense our method can be considered as a simultaneous image restoration and transformation process.

It is straightforward to apply our framework to other applications such as image denoising. For example, we can generate a training set of paired images by adding Gaussian noise to clean images. Coupled dictionaries can be optimally learned to both model the characteristics of noisy images in the source space, and be used to reconstruct target images of high quality. Given a patch of a noisy test image, its denoised version can be estimated by local regression in the learned, coupled sparse feature spaces. With trained dictionaries specifically for image denoising, our method is potential to achieve performance comparable to state-of-the-art techniques based on learned sparse models [25].

2 Related Work

One of the well established image transformation approaches is Freeman *et al.*'s NN example-based learning [4]. Given a test image in the source space, each patch of the test image is compared with the training patches in the source image space, and its several nearest neighbors are selected as candidates. To reconstruct the target image, the candidate patches in the target space are selected and stitched using Markov random fields (MRF). The same approach is also applied to face sketch synthesis [52]. Since this method uses only one of the nearest neighbors for reconstruction, it is susceptible to suffer from overfitting, visually producing noisy and/or

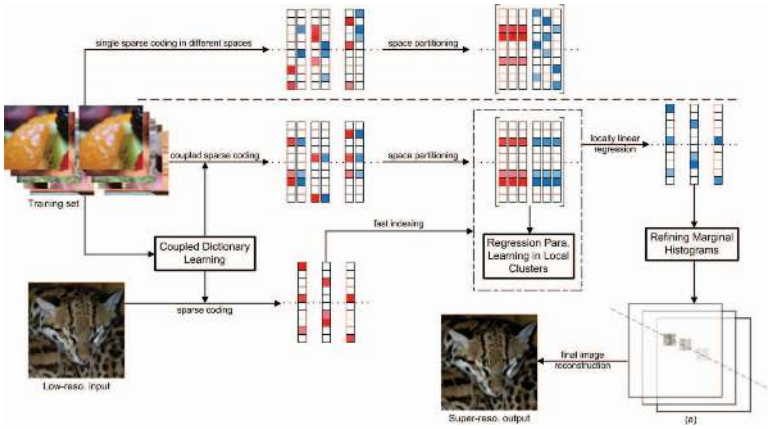


Fig. 1. Pipeline of our proposed framework using super-resolution as an example, which is below the dashed line. For comparison, top of the dashed line illustrates *independent single* sparse coding in different image spaces. Red squares in thin rectangles represent sparse coefficients of image patches in the source space, and blue squares for sparse coefficients of image patches in the target space. Different levels of color transparency represent the order of atoms being selected in sparse coding. Compared with independent single sparse coding, sparse coefficients obtained by coupled space coding are much better aligned. And sparse coefficient vectors in the same local cluster are similar in both coefficient values and patterns of nonzero support. (a) shows multiple constraints for each pixel of the target image to be estimated.

jaggy artifacts. As extensions, neighbor embedding [7] made benefits of multiple NNs, and mixture of mapping experts [11] was learned by locally linear regression.

In terms of performing mappings in projected feature spaces, Lin *et al.* [10] found the most correlative subspaces of different image styles, and performed regression at the reduced subspace dimensions. For each style, it could find only one common subspace, limiting its applicability to only mapping of face images with regular structure. Li and Adelson [9] computed image wavelet coefficients, and considered "nested binning" subband coefficients of local neighborhood centered at each training image pixel. However, nested binning simplified high-dimensional space partitioning as *independent* partitioning along each feature dimension (subband), and thus produced lots of empty hyper-rectangular bins.

Yang *et al.* [15] used sparse coding for super-resolution, which enforces corresponding low- and high-resolution image patches to share the same sparse feature values. However, this is hard to achieve, even with adapted dictionaries. In practice, either more nonzero elements in a sparse feature vector, or dictionaries of many times over-completeness are required. In [15], approximately 10 times over-complete dictionaries were actually used, and this increases the computation cost of sparse coding. Using sparse representation with more nonzero elements (i.e., non-sparse representation) tends to generate undesired noisy patches, as verified in our super-resolution experiments in Section 8. It is not clear either how their method works on other types of image transformation. In fact, we have conducted experiments on intrinsic image estimation using the method in [15]. Results reported in Section 7 show that it fails at such a general type of image transformation application. Compared with [15], our method allows the sparse feature vectors of corresponding patches to choose different values, which also leads to a sparser representation than [15] does. Thus our method has more power and flexibility to describe different image spaces.

Mairal *et al.* [36] also considered dictionary learning for image transformation. They proposed a supervised learning method to learn a dictionary in the source image space and a corresponding transformation matrix. The learned *global* transformation matrix was used to map sparse features of source image patches to intensity values of target patches. Instead, our method is based on a local regression approach that learns transformation parameters for each of a great number of local clusters. Consequently our method admits a much richer model, resulting in improved mapping power than the global method used in [36]. Although supervised learning may result in a dictionary better adapted to the task at hand, the used global transformation model limits its applicability to only image restoration applications, e.g., a classical inverse halftoning problem as considered in [36]. It is unclear how their method can perform on more general image transformation applications.

Our method also falls in a broader category of piecewise linear estimation (PLE). A popular choice of PLE is based on Gaussian Mixture Models (GMM) [46], which describes local image patches with a mixture of Gaussian distributions. A key step in GMM model learning is to divide the data (image patches) into precise and representative local clusters. This might be applicable to image restoration problems where it could be case that only patches of the observed (degraded) image itself are involved in clustering. This is exactly the approach taken in [47], where maximum a posteriori expectation-maximization (MAP-EM) algorithm was used for the clustering purpose. However, as we discussed in Section

1, for general image mapping problems using a large training set of paired images, it is very difficult to achieve precise clustering which requires a huge number of local neighborhoods. Similar to k-means clustering, in such a scenario the MAP-EM algorithm considered in [47] could be prohibitively slow. For these methods, it also requires high computational complexity to retrieve local clusters (Gaussians) in the estimation process. Instead, our method uses coupled sparse coding to learn a sparse feature space, based on which an efficient partition scheme is proposed to divide the space into local clusters, i.e., a union of low-dimensional linear subspaces. It also facilitates efficient retrieval of closest local clusters. Reported experiments in Section 5 show the efficacy and efficiency of our proposed subspace partition scheme.

Sparse Coding Given a signal vector $\mathbf{x} \in \Re^N$ and a dictionary $\mathbf{D} \in \Re^{N\times K}$ whose columns are dictionary atoms, sparse coding finds a sparse coefficient vector $\mathbf{a} \in \Re^K$ by solving $\min_{\mathbf{a}} \|\mathbf{x}-\mathbf{D}\mathbf{a}\|_2^2 \;\; s.t. \;\; \|\mathbf{a}\|_0 \leq L$, where $\|\cdot\|_0$ is the l_0 norm, counting the nonzero elements of $\mathbf{a}$. $\mathbf{D}$ is over-complete when $K > N$. Each column of $\mathbf{D}$ is constrained to have unit l_2 norm. The l_0 norm above can be replaced with l_1 regularization [42], [43], resulting in the convex sparse coding problem $\min_{\mathbf{a}} \|\mathbf{x}-\mathbf{D}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1$, where λ balances the sparsity and reconstruction error. $\mathbf{D}$ can be chosen as pre-defined ones, or be adapted to training data by learning. Representative dictionary learning algorithms include K-SVD [32], Fields of Experts (FoE) [27], and others [26], [33].

3 Framework Overview

Given an input source image $\mathbf{X}$, the task is to infer the target image $\mathbf{Y}$ based on $\mathbf{X}$. This is equivalent to maximizing the probability $p(\mathbf{Y}|\mathbf{X})$. If we model the inference of $\mathbf{Y}$ on a MRF, the Hammersley-Clifford theorem tells that we can write the probability as a Gibbs distribution $p(\mathbf{Y}|\mathbf{X}) \propto \exp(-\sum_\omega \Phi_\omega(\mathbf{y}_\omega|\mathbf{X}))$, where $\Phi_\omega(\mathbf{y}_\omega|\mathbf{X})$ is the potential function for a local patch (clique) $\mathbf{y}_\omega$ and conditioned on input $\mathbf{X}$. In other words, $p(\mathbf{Y}|\mathbf{X}) \propto \prod_\omega p(\mathbf{y}_\omega|\mathbf{X})$, where $p(\mathbf{y}_\omega|\mathbf{X})$ is related to $\Phi_\omega(\mathbf{y}_\omega|\mathbf{X})$ by an exponential. We make additional assumption that the estimation of $\mathbf{y}_\omega$ only depends locally on its corresponding patch $\mathbf{x}_\omega$ of $\mathbf{X}$ (patch size of $\mathbf{x}_\omega$ could be the same or larger than that of $\mathbf{y}_\omega$), i.e., $p(\mathbf{y}_\omega|\mathbf{X}) = p(\mathbf{y}_\omega|\mathbf{x}_\omega)$.

3.1 Local Parametric Regression

Natural images are high-dimensional and statistically non-Gaussian. It is in general difficult to directly model $p(\mathbf{y}_\omega|\mathbf{x}_\omega)$. We consider to reduce the problem by dividing training patch samples into local clusters, and then perform parametric regression within each cluster. Given M training patch pairs $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$ with $\mathbf{x}^i \in \Re^{N_x}$ and $\mathbf{y}^i \in \Re^{N_y}$, suppose we can divide the set into C local clusters. Each cluster $c \in \{1, \ldots, C\}$ contains $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^{M_c}$. We model $p(\mathbf{y}|\mathbf{x})$ for each subset $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^{M_c}$. However, as explained in Section 1, it is difficult to obtain precise local neighborhoods by space partitioning over a huge set of patch samples in the raw intensity domain. On the other hand, natural images are inherently sparse signals. For image patches in $\Re^{N_x}$ and $\Re^{N_y}$, they generally do not scatter uniformly in the space. For example, spatially smooth patches, edges, and textures are the most common patterns observed in natural images. To model $p(\mathbf{y}|\mathbf{x})$, it is typical to use product of expert distributions in some transform domains, where experts are usually defined on one-dimensional subspaces and their distributions are shown to be highly kurtotic.

In fact, sparsity is one of the critical reasons for the popularity of transform domain image processing. In this work, we push the step further by learning transform domain sparse representations in both the source and target image spaces. Sparse coding of source image patches can be seen as a nonlinear feature selection process, which can identify the most significant features (salient structures) of image patches for clustering and local regression. When reconstructing target images using sparse coefficients over optimally learned target space dictionary, it is possible to produce natural images of higher quality, with the effect of noise removal and missing data recovery.

Moreover, to facilitate image mappings by local parametric regression, we require sparse representations in the source and target image spaces to be coupled. That is, the sparse coefficient vectors of a pair of corresponding source and target image patches have the same support. Suppose $\mathbf{D}_x \in \Re^{N_x\times K}$ and $\mathbf{D}_y \in \Re^{N_y\times K}$ are the learned dictionaries for source and target spaces respectively, we compute sparse feature vectors $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$ of the set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$ over $\mathbf{D}_x$ and $\mathbf{D}_y$ via coupled sparse coding. Based on $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$, we devise an efficient partitioning scheme in the sparse feature spaces, which can result in local clusters with $\{\mathbf{a}_x^i\}_{i=1}^{M_c}$ in any cluster c being similar in both coefficient values and patterns of nonzero support. As a result, $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^{M_c}$ are also similar in the pattern of nonzero support due to coupled sparse coding, as shown in Figure 1. The problem of image transformation by local parametric regression becomes to learn $p(\mathbf{a}_y|\mathbf{a}_x)$ for each of the local clusters. If we model $p(\mathbf{a}_y|\mathbf{a}_x)$ within the c^{th} cluster as Gaussian, regression amounts to learning a mapping function $f(\mathbf{a}_x, \mathbf{W}_c) = \mathbf{W}_c\mathbf{a}_x$ by solving a least squares problem

$$\min_{\mathbf{W}_c} \|\mathbf{W}_c\mathbf{A}_x^c - \mathbf{A}_y^c\|_F^2, \tag{1}$$

where $\mathbf{A}_x^c = [\mathbf{a}_x^1, \mathbf{a}_x^2, \ldots, \mathbf{a}_x^{M_c}] \in \Re^{K\times M_c}$, $\mathbf{A}_y^c = [\mathbf{a}_y^1, \mathbf{a}_y^2, \ldots, \mathbf{a}_y^{M_c}] \in \Re^{K\times M_c}$, and $\mathbf{W}_c \in \Re^{K\times K}$ contains the parameters to be learned.

3.1.1 Benefits of Coupled Sparsity

For image transformation based on local parametric regression, coupled sparse coding has important advantages, such as less memory cost and reliability of regression parameter learning. In particular, equation (1) is equivalent to learning K mapping functions for all the feature dimensions in the target space

$$\min_{\mathbf{w}_c^k} \|\mathbf{A}_x^{c\top}\mathbf{w}_c^k - \mathbf{a}_y^k\|^2, \quad \forall k \in \{1, \ldots, K\}, \tag{2}$$

where $\{\mathbf{w}_c^{k\top}\}_{k=1}^K$ and $\{\mathbf{a}_y^{k\top}\}_{k=1}^K$ respectively correspond to the row vectors of $\mathbf{W}_c$ and $\mathbf{A}_y^c$. However, since $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^{M_c}$ have the property of coupled sparsity, without loss of generality we can assume each vector in $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^{M_c}$ has $L \ll K$ nonzero elements at the same coordinates. Thus in practice, we only need to learn L mapping functions corresponding to those feature dimensions with nonzero elements in $\{\mathbf{a}_y^i\}_{i=1}^{M_c}$. And for each mapping function $f(\cdot, \mathbf{w}_c^k)$ with $\mathbf{w}_c^k \in \Re^K$, only L effective parameters are to be estimated. Compared with $\mathbf{W}_c \in \Re^{K\times K}$, the number of parameters to be learned is much smaller. This is beneficial for both ease of learning and reducing memory cost.

When coupled sparse coding is not enforced, there will be no such benefits. Suppose we perform *independent* sparse coding for source and target training samples respectively, and training sample pairs are divided into local clusters based on obtained sparse feature vectors $\{\mathbf{a}_x^i\}_{i=1}^M$ in the source space. In any cluster c, although the columns of $\mathbf{A}_x^c$ can be similar in both coefficient values and support patterns, the support patterns for each pair $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}$ are in general different (as shown in the top of Figure 1), i.e., the columns of $\mathbf{A}_y^c$ have different support patterns. As a consequence, it is necessary to learn K mapping functions via solving (2) for all the feature dimensions in the target space. Compared with coupled sparsity, there are K/L times more parameters to learn and correspondingly a greatly increased amount of memory cost. Since mapping functions are learned for all the feature dimensions, for any test patch $\mathbf{x}'$ with its sparse vector $\mathbf{a}'_x$, the estimation of target space $\hat{\mathbf{a}}'_y$ is in general not sparse for a desired image reconstruction.

When sparse coding is not coupled, it is even worse that modelling $p(a_y^k|\mathbf{a}_x)$ as Gaussian becomes inappropriate for any feature dimension $k \in \{1, \ldots, K\}$, and correspondingly learning mapping relations based on least squares regression is unreliable. For training patches $\{\mathbf{y}^i\}_{i=1}^{M_c}$ in the c^{th} cluster, assume $\mathbf{d}_y^k$ is a dictionary atom in the target space and corresponds to some salient structure shared by $\{\mathbf{y}^i\}_{i=1}^{M_c}$. Ideally most of these patches should fire on $\mathbf{d}_y^k$ via sparse coding in the target space, which means that most coefficients of $\mathbf{a}_y^{k\top}$ (the row vector of $\mathbf{A}_y^c$ corresponding to $\mathbf{d}_y^k$) should have relatively large magnitude. However, due to sensitiveness of sparse coding process [37] and dictionary over-completeness, it is often the case that very similar patch structures may be quantized on different dictionary atoms. Thus in general much fewer patch samples in $\{\mathbf{y}^i\}_{i=1}^{M_c}$ may fire on $\mathbf{d}_y^k$, resulting in a $\mathbf{a}_y^{k\top}$ with many elements of zero value, which is indeed unreliable for learning of $\mathbf{w}_c^k$ by solving a least squares problem as in (2). While modeling $p(a_y^k|\mathbf{a}_x)$ as Gaussian is inappropriate, it is also difficult to find other generally suitable conditional relations for this learning task.

One may be also interested in directly estimating raw intensity values of target patches. This can be realized by changing (2) slightly as $\min_{\mathbf{w}_c^k} \|\mathbf{A}_x^{c\top}\mathbf{w}_c^k - \mathbf{y}^k\|^2$ for $k \in \{1, \ldots, N_y\}$, where $\{\mathbf{y}^{k\top}\}_{k=1}^{N_y}$ correspond to row vectors of data matrix $\mathbf{Y}^c = [\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^{M_c}]$ of target patches. Therefore we need to learn N_y mapping functions for all pixels inside the target patch. In general, N_y is much larger than L. Since natural images are signals with structures, learning the mapping function for each pixel of a patch is obviously unnecessary. Estimating raw intensities directly has other drawbacks when training patch samples are corrupted by noise or missing data, which may degrade the learning of mapping parameters. Indeed, when training samples are limited, corrupted training data may bias the learning process, making it more difficult to form an accurate prediction model [48]. Learning between coupled sparse representations is less affected in such situations as sparse coding of target patches has the inbuilt property of robustness against data corruption. We present comparative experiments in Section 7 to show the superiority of coupled sparsity over the above mentioned alternatives.

3.2 Reconstruction of the Target Image

Independently estimated sparse coefficients of local patches may not be consistent as a whole image with desired statistical properties in the target image space. [1] These statistical properties can be seen as prior knowledge and can be learned from training images, as [27] did. In this work, we do not aim to learn those priors and to do a MAP inference. Instead, we take an empirical approach to refine the marginal histograms of estimated sparse coefficients. Specifically, for each feature dimension $k \in \{1, \ldots, K\}$, denote $\mathcal{H}_y^k$ as the marginal histogram of the estimated coefficient values $\{a_y^{k,\omega}\}$ of local patches positioned at each ω in the target image $\mathbf{Y}$, corresponding to the atom $\mathbf{d}_y^k$ of target dictionary $\mathbf{D}_y$. And $\widehat{\mathcal{H}}_y^k$ is a reference marginal histogram for $\mathbf{d}_y^k$ in the target space. To refine $\mathcal{H}_y^k$, we apply histogram matching [3], which matches $\mathcal{H}_y^k$ against $\widehat{\mathcal{H}}_y^k$ by the function

$$\mathcal{F}(a_y^{k,\omega}) = \widehat{\mathcal{C}}_y^{k^{-1}}\big[\mathcal{C}_y^k(a_y^{k,\omega})\big], \tag{3}$$

where $\mathcal{C}_y^k$ and $\widehat{\mathcal{C}}_y^k$ are cumulative histograms defined by $\mathcal{C}(a) = \int_{-\infty}^{a} \mathcal{H}$. To find $\widehat{\mathcal{H}}_y^k$, we compare marginal histograms of the input source image $\mathbf{X}$ with those of each source image in the training set using KL divergence. Then marginal histograms of the corresponding target images of those found nearest neighbors are averaged to give the reference histograms.

When local patches are densely overlapped, estimated sparse coefficients together with the learned dictionary

1. Since we have modeled $p(\mathbf{a}_y|\mathbf{a}_x)$ within each local cluster as a Gaussian, by doing regression we actually estimate a conditional mean of the sparse coefficients of the target patch. But probability of image patches may not be well modeled as a Gaussian. From another perspective, the whole image as an ensemble of local patches follows some prior distribution. If we view target space dictionary atoms as learned linear filters, the prior $p(\mathbf{Y})$ can be expressed as $\prod_\omega \prod_k p(\mathbf{d}_y^{k\top}\mathbf{y}_\omega) = \prod_\omega \prod_k p(a_y^{k,\omega})$, where $a_y^{k,\omega}$ denotes the filter response and is equivalent to elements of sparse coefficient vector $\mathbf{a}_y$ of patch $\mathbf{y}_\omega$.

provide multiple constraints for each pixel of the target image, as shown in Figure 1-(a). To reconstruct the target image, we use different alternatives in applications of intrinsic image estimation and image super-resolution. Details will be explained in Sections 7 and 8.

4 Coupled Dictionary Learning

Given the training set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$ of paired patches, coupled sparse coding amounts to finding sparse feature vectors $\mathbf{a}_x \in \Re^K$ and $\mathbf{a}_y \in \Re^K$ over dictionaries $\mathbf{D}_x \in \Re^{N_x \times K}$ and $\mathbf{D}_y \in \Re^{N_y \times K}$, to produce good approximation of any training patch pair, while enforcing the pair $\{\mathbf{a}_x, \mathbf{a}_y\}$ to share a common support. Define $\mathbf{A} = [\mathbf{a}_x, \mathbf{a}_y] \in \Re^{K \times 2}$, the problem can be stated as

$$\min_{\mathbf{A}} \frac{1}{2}\big(\|\mathbf{x} - \mathbf{D}_x\mathbf{a}_x\|_2^2 + \|\mathbf{y} - \mathbf{D}_y\mathbf{a}_y\|_2^2\big) + \lambda\|\mathbf{A}\|_{p,q} \qquad (4)$$

where $\|\mathbf{A}\|_{p,q} = \sum_{k=1}^K (\|\mathbf{a}_k^\top\|_q)^p$ is some matrix norm with parameters p, q properly chosen to induce coupled solution sparsity, and $\mathbf{a}_k$ is the kth row of $\mathbf{A}$ with $\mathbf{a}_k = [a_x^k, a_y^k]$. λ is a regularization parameter. Parameters p, q can be chosen as $0 \le p \le 1$ and $q \ge 1$. When $p = 1$, it penalizes the nonzero rows of $\mathbf{A}$, thus promoting a common sparsity pattern between $\mathbf{a}_x$ and $\mathbf{a}_y$. Typical choices of q are 2 and ∞. In this work, we use $q = 2$ for its simplicity, and also for an efficient algorithm presented shortly. Thus we have $\|\mathbf{A}\|_{1,2} = \sum_{k=1}^K \|\mathbf{a}_k^\top\|_2$.

We perform coupled dictionary learning on the large training set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$. Upon convergence, we expect the learned atoms of dictionaries $\mathbf{D}_x$ and $\mathbf{D}_y$ can model both the coherent structures of each individual image space and correlation characteristics between different spaces, and thus can facilitate better couplings of sparse coding. In particular, coupled dictionary learning amounts to solving the objective function

$$\begin{aligned}\min_{\{\mathbf{A}^i\},\mathbf{D}_x,\mathbf{D}_y} \sum_{i=1}^M \frac{1}{2}\big(\|\mathbf{x}^i - \mathbf{D}_x\mathbf{a}_x^i\|_2^2 + \|\mathbf{y}^i - \mathbf{D}_y\mathbf{a}_y^i\|_2^2\big) \\ +\lambda\sum_{k=1}^K \|\mathbf{a}_k^{i\top}\|_2 \\ s.t.\ \|\mathbf{d}_x^k\|_2^2 \le 1,\ \|\mathbf{d}_y^k\|_2^2 \le 1 \quad \forall\, k = 1, \ldots, K. \end{aligned} \qquad (5)$$

The above optimization is not convex with respect to $\mathbf{D}_x$ or $\mathbf{D}_y$. However, it is a joint optimization problem, and is convex with respect to $\mathbf{D}_x$ or $\mathbf{D}_y$ (or, $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$) while holding $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$ (or, $\mathbf{D}_x$ and $\mathbf{D}_y$) fixed. This suggests an iterative approach that alternates between the *coupled sparse coding* stage (solving $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$) and the *dictionary update* stage (updating $\mathbf{D}_x$ and $\mathbf{D}_y$). Similar strategy has been employed for single space dictionary learning problems [32], [33].

Given fixed dictionaries $\mathbf{D}_x$ and $\mathbf{D}_y$, solving $\{\mathbf{a}_x, \mathbf{a}_y\}$ (or $\mathbf{A}$) in equation (4) is equivalent to a regularized least squares regression problem with a non-differentiable sparsity inducing term $\|\mathbf{A}\|_{1,2}$. In literature, *block coordinate descent* has been used for related group Lasso problems in case of a single dictionary [28]. In this work, we use an active set method based on the *block coordinate gradient descent* algorithm [30], namely active set BCGD algorithm. Our algorithm maintains an active set of growing size and systematically searches for the optimal active set and sparse solutions, similar to the strategy used in [29] and reviewed in [44]. It is particularly efficient by taking the advantage of the high degree of solution sparsity. To enforce optimality inside the active set, we use BCGD, while [29] is based on a projected gradient method that requires specification of a tuning parameter to ensure convergence. In contrast, our algorithm based on BCGD is fully automatic and with proved convergence. For details of our active set BCGD algorithm, interested readers may refer to Appendix A in the supplemental material.

Given coupled sparse solutions $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$, (5) boils down to a constrained least squares problem. We adopt Lee *et al.*'s Lagrange dual method [33], which is efficient as it involves much fewer optimization variables than the primal problem. Details of the dictionary learning algorithm and its convergence analysis are given in Appendices B and C in the supplemental material.

4.1 Solution Paths of Training Patch Samples

Equation (4) pursuits a coupled sparse solution given a fixed value of λ. In some situations, it is useful if we can compute the set of solutions for all possible λs, i.e., the solution path. For example, by tracing the solution path of sparse coding, one can identify the order of dictionary atoms being sequentially selected. For group Lasso, this is not as easy as the standard Lasso, e.g., the LARS [42], and costs large additional computation, as the solution path of group Lasso is not piecewise linear. However, active set BCGD algorithm can approximate the solution path on a grid of λ values with little additional cost. Upon the solution at a certain λ value, the solution at the next close-by λ value can be efficiently found using the earlier solution as a *warm start*, and typically only several further iterations are needed to get the new solution.

Computing solution paths of training patch samples is particularly useful in our problem setting. In (4), we use λ to control the level of coupled solution sparsity. During test stage, we are actually solving a *single* sparse coding (standard Lasso) problem

$$\min_{\mathbf{a}'_x} \frac{1}{2}\|\mathbf{x}' - \mathbf{D}_x\mathbf{a}'_x\|_2^2 \quad s.t. \quad \|\mathbf{a}'_x\|_1 \le t \quad , \qquad (6)$$

where t is a constraint value for the sparse solution of input patch $\mathbf{x}'$. For image transformation, it is necessary to require that the test patch solutions and those of training patches in the source space be at the same sparsity level, i.e., $\|\mathbf{a}'_x\|_1 = \|\mathbf{a}_x\|_1$. In practice, we address this issue by specifying a shared constraint value t for both training and test sparse coding. For training, we compute approximate solution path and tune λ until $\|\mathbf{a}_x\|_1 \approx t$. For testing, we choose LARS algorithm to solve (6). It should be noted that for a source patch $\mathbf{x}$, computing its sparse solutions $\mathbf{a}_x$ using (4) and $\mathbf{a}'_x$ using (6) cannot

guarantee that $\mathbf{a}'_x = \mathbf{a}_x$. Even though we can enforce $\|\mathbf{a}'_x\|_1 = \|\mathbf{a}_x\|_1$, they may still have different nonzero support patterns. Our coupled dictionary learning can relax this problem. After the coupled dictionary learning process converges, the learned corresponding dictionary atoms between source and target spaces will be aligned and correlated to each other. As a consequence, most of patch samples will have the same first several selected nonzero indices by using (4) and (6), as verified by experiments in Appendix C in the supplemental material. For a test patch $\mathbf{x}'$ with its sparse solution $\mathbf{a}'_x$ computed by (6), it will index into the right local cluster with training patches very similar to $\mathbf{x}'$, based on our proposed space partition method in Section 5. Consequently an accurate mapping of $\mathbf{x}'$ to its target patch $\mathbf{y}'$ can be obtained. Experiments on intrinsic image estimation and super-resolution show that this scheme works well in practice.

5 Partitioning in the High-Dimensinal Sparse Feature Space

Our image transformation method is based on local parametric regression. Given a training set of paired patches, local regression requires first dividing the training samples into local clusters so that each cluster contains samples which are as similar as possible, and the number of samples is large enough to learn the regression parameters reliably. Given a query source patch, such a data dividing should also facilitate efficient retrieval of the cluster closest to the query. This is essentially a high-dimensional space partitioning problem.

Space partitioning is closely related to the nearest neighbor (NN) search. Classical approaches such as kd-tree are only efficient in multi-dimensional spaces. Approximate nearest neighbor (ANN) search [22] can perform efficient search up to several dozen dimensions. Locality-sensitive hashing (LSH) [23] has been shown to be effective in higher dimensions. Due to its randomness, the codes generated in LSH are in general non-compact, which limits its practicality in large-scale search problems. Other empirical methods such as spectral hashing [24] can achieve code efficiency. However, they cannot generally produce balanced buckets (clusters), i.e., each bucket contains similar number of samples, which is desired for local regression based image transformation.

Besides data structures, search quality and efficiency critically depend on the concise feature representation of signals and the proximity measure. Our image transformation is based on the learned sparse feature representation. While the sparse representation of patch samples is in high dimensions, those nonzero features are very sparse and essentially correspond to salient patch structures identified by corresponding dictionary atoms. When devising a data structure for efficient retrieval of the closest cluster for a query patch, it is beneficial to take advantage of such a high-dimensional but sparse feature representation of signals. Besides sparsity, we have additional information of the dimension selection order in sparse coding of image patches via solving (4) and (6). While most of existing methods use Euclidean distance for similarity search, we argue that the dimension selection order in sparse vector representation also provides important criteria for matching similar patches.

For reducing memory cost and ease of parameter learning in our local regression approach, it is desirable that after space partitioning, sparse feature vectors of paired patch samples within each cluster have the same support (cf. Section 3.1.1). In this work, we devise a data indexing structure to specifically meet this requirement. Our scheme iteratively use two kinds of information, namely the dimension selection order in sparse feature vector and the coefficient values on the selected feature dimensions, to partition the space into cells (clusters).

Problem Definition Given M paired patches $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$, their sparse feature vectors $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$ are computed via coupled sparse coding over $\mathbf{D}_x$ and $\mathbf{D}_y$, where $\mathbf{D}_x$ has K atoms and thus $\mathbf{a}_x^i \in \Re^K$. Perform a partitioning of the feature space $\Re^K$ so that given a query sparse vector $\mathbf{a}'_x$ of any test patch $\mathbf{x}'$, a cell of the partition as close to $\mathbf{a}'_x$ as possible can be efficiently found. We define samples in $\Re^K$ as close if their orders of feature dimensions being selected in sparse coding are the same, and their feature values measured by Euclidean distance are similar.

5.1 The Data Indexing Structure

Briefly speaking, we are interested in quantizing the space $\Re^K$ into cells so that each cell can be represented as a compact code and indexed efficiently. The efficiency comes from table look-ups based on the quantized code representations of local cells. Besides, each cell should also contain enough samples for reliable learning of local regression parameters.

Two distinctive quantization functions are defined under our scheme. Based on the information of dimension selection order in sparse coding, our first quantization function is defined as

$$q_o^j(\mathbf{a}_x) = z, \quad z \in \{0, \ldots, K-1\}, \tag{7}$$

where K is the dimensionality of the feature space, and j corresponds to the dimension selection order. For example, when $j = 1$, $q_o^1(\mathbf{a}_x)$ identifies the dimension of $\mathbf{a}_x$ that is firstly selected in sparse coding of $\mathbf{x}$. For each order of j, we can maximally divide the set of samples into K subsets, and the number of bits needed to encode the data is no more than $\log_2^K$. Our second quantization function is a 1D quantizer. It concerns with partitioning a selected dimension of $\Re^K$ based on feature values of samples on that dimension. Defined as

$$q_v(a_x) = z, \quad z \in \{0, \ldots, \eta-1\}, \tag{8}$$

we require that after quantization, each of the η buckets contains roughly the same number of samples. The number of bits needed for this quantization is $\log_2^\eta$. The introduction of this 1D quantizer is essential to realize a balanced partition for local regression based image transformation, as will be discussed shortly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

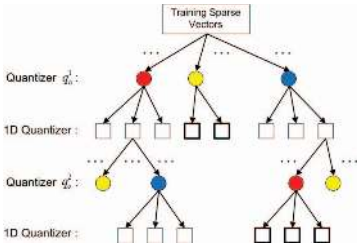


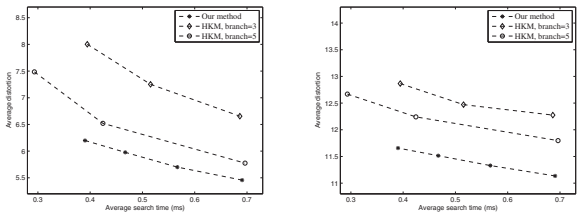
Fig. 2. Partitioning in the high-dimensional sparse feature space, using the information of both sparse feature values and the order of feature dimensions being selected. Color circles represent partitioned subsets based on dimension selection order, different colors for different feature dimensions. Squares represent splitted buckets based on feature values by 1D quantization along currently selected feature dimension. Black squares represent leaf cells.

Given the training set $\{\mathbf{a}_x^i\}_{i=1}^M$, we first use quantizer q_o^1 to divide the M samples into K subsets, each of which contains samples with the same dimension being firstly selected in sparse coding of them. For those samples contained in any z^{th} subset, we then apply quantizer q_v on them by examining their feature values of dimension z, yielding η buckets, each of which roughly has the same number of samples. We store the values of bucket boundaries on dimension z for later query use. Such 1D quantization can be easily obtained via order statistics. Up to now we have partitioned the M samples into $K \times \eta$ cells. Each cell can be encoded using $\log_2^{K\eta}$ number of bits. At the second level, for each of the $K\eta$ cells obtained, we further partition it using quantizer q_o^2 and then q_v. The process continues by iteratively using these two quantization functions, until the leaf cells contain a specified minimum number (denoted as ζ) of samples, which form our local clusters and can be used for local parametric regression. Figure 2 illustrates the construction of our indexing structure. For each local cell, we also store the mean values of its examined feature dimensions (i.e., the centroid in the examined subspace) for later query use. Assuming at most d dimensions have been examined to reach leaf cells, we thus partition the space $\Re^K$ into a number of cells bounded by $K^d\eta^d$, and each cell can be encoded using no more than $d\log_2^{K\eta}$ bits. For efficient retrieval, a look-up table is constructed based on quantized code representations of local cells.

Query Procedure Given the sparse vector $\mathbf{a}'_x$ of a query patch $\mathbf{x}'$, we first compute its code representation by iteratively using the quantizers q_o^j and q_v to locate the first candidate cell, whose code is also the code of query $\mathbf{a}'_x$. In this process, we maintain a dynamic index of intermediately retrieved cells. More specifically, given the dimension selection order of $\mathbf{a}'_x$, we can easily obtain $q_o^1(\mathbf{a}'_x) = z$ assuming z is the first dimension being selected in sparse coding of $\mathbf{x}'$. We can then get the index of those cells whose code given by q_o^1 is also z, and also the bucket boundaries for use of q_v on dimension z of $\mathbf{a}'_x$. After getting the code from 1D quantizer, we further reduce the maintained index of retrieved cells. The dynamic retrieval of cells and bucket boundaries for 1D quantization can be efficiently implemented using code collision and table look-up. This process guarantees that the found cell has the same dimension selection order as that of query. However, such cells are not unique, and the first candidate cell is not necessarily closest to the query in terms of feature value distance (distances of query to cell centroids). At the second stage, we compute and compare feature value distances between the query and those retrieved cells with the same dimension selection order, and find the closest one as the cell that $\mathbf{a}'_x$ is finally located.

Properties of Our Scheme Compactness of data representation is a key factor in large-scale efficient search. In our scheme, the number of total cells is bounded by $K^d\eta^d$, which grows exponentially with d. Fortunately, sparse representation of patch samples has the property that feature values on the first several selected dimensions in sparse coding can dominate the signal energy, thus we only need to examine several dimensions to partition the space, i.e., d is very small ($d = 2 \sim 4$ in practice). The number of total cells obtained is also much fewer than $K^d\eta^d$. When K is smaller than 256 and η is around 15, the code length of 48 bits is enough to represent the data. On another hand, image transformation based on local regression requires a balanced partition of the training samples. Existing space partitioning methods for sparse high-dimensional data, such as inverted files structure used in image retrieval [38], [39], cannot apply here. Indeed, space partitioning using only the support information of sparse representation is a combinatorial problem, yielding different groups of samples either too big or too small. Instead, we realize a balanced partition by iteratively using the information of dimension selection order (support pattern) via the quantization function q_o^j, and feature values via the 1D quantizer q_v. The parameter η in 1D quantization plays an important role here. When η is larger, we rely more on feature values. Conversely we rely more on the dimension selection order information.

Analysis of Search Complexity Our search procedure involves retrieving local cells with the same dimension selection order as that of query, and also retrieving bucket boundaries used by 1D quantizer, both of which can be performed in constant time based on table look-up. For each dimension being examined, the 1D quantizer q_v takes $\mathcal{O}(\eta)$ time when linear scan is used. Assume d dimensions have been examined to find the first candidate cell, the time cost is $\mathcal{O}(d\eta)$. At the second query stage, we compute Euclidean distances between the query and the centroids of those retrieved cells with the same dimension selection order. Denote the number of these cells as N_o. At the extreme case, N_o could be as big as η^d. However, in practice, cells with the same dimension selection order are much fewer, and it is a few dozen on average. Thus the total time complexity for a query is $\mathcal{O}(d(\eta + N_o))$.



(a) (b)

Fig. 3. Quality and efficiency comparison with hierarchical k-means (HKM) in both source (a) and target (b) image spaces. Branch factors 3 and 5 are used in HKM. Search time of the closest cluster for a query is measured by millisecond (ms).

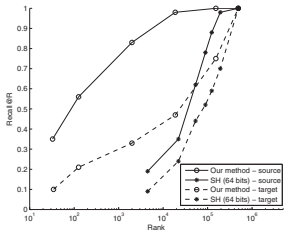


Fig. 4. Comparison of NN retrieval with spectral hashing (SH). SH uses 64 bits. Retrieval quality is measured by recall@R.

5.2 Experimental Verification

We used the MIT intrinsic dataset [16], and randomly sampled $500,000$ patch pairs from grayscale images and their shading components, of which $10,000$ pairs were randomly chosen as the query set, and others were the training set. The patch size was 8×8, and the dictionary size was set as $K = 128$. All comparative experiments were conducted using Matlab implementation on an Intel Core 2 Duo CPU running at 2.53GHz.

The quality of a space partition is commonly measured by its average distortion [2]. We used the Euclidean distance to measure the distortions of query patch samples in the raw intensity domain, although the partition of our method is induced in the sparse feature space, and the partitions of other methods to be compared are obtained using training raw patches. Appendix D in the supplemental material shows how the performance of our method depends on the sparsity level, the value of ζ (minimum number of samples inside each cell), and the value of η for 1D quantizer q_v. In the experiments below, we set $\eta = 15$ and the sparsity level as 8 nonzero elements in sparse vectors of patch samples, while ζ was varying for quality efficiency trade-off.

K-means clustering is based on the optimal space partition criteria (minimum distortion) [54]. However, it is prohibitively costly to apply it on the database as large as half a million samples. We thus compare our method with hierarchical k-means (HKM) [45], [53], which is an efficient improvement over k-means. Except for comparison in the source space, for image transformation, it is also important to investigate the quality of the partition in the target space. In Figure 3, we compare average distortions of different methods against search time in both the source and target image intensity domains, where HKM with branch factors 3 and 5 were used (branch factor is the defined maximal number of clusters at each hierarchical level of HKM). Figure 3 shows that our method is better than HKM in terms of quality efficiency trade-off. At the same distortion level, our method is much faster than HKM.

2. Given a query, average distortion measures the expected distance between the query and the centroid of the cell that the query falls in.

We also test the effectiveness of our method for efficient high-dimensional approximate NN search. We set $\zeta = 20$ and partitioned the database points into $15,000$ cells. Each cell was encoded using 48 bits. For a query point, database points inside each returned cell were considered as retrieved NNs. In ANN search literature, spectral hashing (SH) [24] can produce compact code. We compare our method with SH at 64 bits. To measure the search quality, we use recall@R [3]. While space partitions of different methods are induced in the source space, we measure recall@R in both the source and target intensity domains. Figure 4 plots recall@R over a range of rank R, which shows that our method for NN retrieval is more effective than SH in both source and target spaces. It should be noted that our method is more efficient for retrieval at low rank values, which correspond to the several closest cells for a query point. For example, SH gets recall@R $= 0.19$ in the source space using 154.7 ms for $R = 4500$, while our method gets recall@R $= 0.35$ in the source space using 0.66 ms for $R = 30$.

6 Summary of Algorithmic Procedure

At the training stage, given the training set $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$, we learn coupled dictionaries $\mathbf{D}_x$ and $\mathbf{D}_y$ by solving (5). We then perform path-following coupled sparse coding to get $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$ for $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^M$, by solving (4) with varying parameter λ for each $i \in \{1, \ldots, M\}$. Based on $\{\mathbf{a}_x^i, \mathbf{a}_y^i\}_{i=1}^M$ space partitioning in the sparse feature space is then applied, for which we iteratively use quantization functions (7) and (8) to produce code representations of local cells. Suppose C local cells are obtained, regression parameters $\mathbf{w}_c^k$ inside each cell $c \in \{1, \ldots, C\}$ are then learned, by solving (2) for those feature dimensions $k \in \{1, \ldots, K\}$ with nonzero feature values.

At the test stage, given an input image $\mathbf{X}$, we first extract densely overlapped local patches $\{\mathbf{x}_\omega\}$ from $\mathbf{X}$. For each $\mathbf{x}_\omega$, we perform path-following sparse coding by solving (6), and obtain sparse representation $\mathbf{a}_x^\omega$. To estimate its target sparse representation $\mathbf{a}_y^\omega$, we need to retrieve its closest local cell. To do so we compute the quantized code representation of $\mathbf{a}_x^\omega$ by iteratively using quantizers (7) and (8). By the same process we can locate the first candidate cell, and those cells with

3. recall@R is defined as the probability that the nearest neighbor of a query is among the retrieved R points, averaged over a large number of queries, where distance measure is based on l_2 norm.

the same dimension selection order as that of $\mathbf{a}_x^\omega$. We then compute feature value distances between $\mathbf{a}_x^\omega$ and those retrieved cells, and find the closest one as the finally retrieved cell for $\mathbf{a}_x^\omega$. Suppose it is the c^{th} cell, then each nonzero coefficient $a_y^{k,\omega}$ on dimension k can be simply computed as $\mathbf{w}_c^{k\top}\mathbf{a}_x^\omega$, and we thus obtain the estimated $\mathbf{a}_y^\omega$. After obtaining $\{\mathbf{a}_y^\omega\}$ for all extracted patches $\{\mathbf{x}_\omega\}$, we apply histogram matching using (3) to refine the marginal statistics of $\{\mathbf{a}_y^\omega\}$. Each corresponding patch in the target image $\mathbf{Y}$ can be finally computed as $\mathbf{y}_\omega = \mathbf{D}_y\mathbf{a}_y^\omega$. The estimated densely overlapped patches $\{\mathbf{y}_\omega\}$ provide multiple constraints for each pixel of $\mathbf{Y}$. To reconstruct $\mathbf{Y}$, we use different approaches in intrinsic image estimation and super-resolution. Details will be presented in the following sections.

For a test image with a typical size of 200×300 pixels, if we use 8×8 patches with 5 pixels overlap, it takes 41 seconds on average to perform image transformation, using Matlab implementation of our algorithms on an Intel Core 2 Duo CPU running at 2.53GHz. The main computation comes from sparse coding of local patches, for which we have used a 2 times over-complete dictionary and the LARS algorithm with 12 nonzero elements in the corresponding sparse feature vectors.

7 Intrinsic Image Estimation

We consider a type of intrinsic image estimation that is to decompose an image $\mathbf{I}$ into its *shading* component $\mathbf{S}$ and *reflectance* component $\mathbf{R}$: $\mathbf{I}(x) = \mathbf{S}(x)\mathbf{R}(x)$ with x for a pixel. This is an ill-posed problem. Existing works are either based on a single grayscale image [17], [11], or additional information such as color or multiple images [18], [19], [20]. For the first category, Retinex [17] assumed that small gradients of the observed image correspond to smooth illumination changes of the surface and large gradients correspond to albedo changes. This simple assumption does not hold for real-world images. Tappen *et al.* [11] proposed a nonlinear regression approach to estimate gradients of intrinsic images. For the second category, Shen *et al.* [18] used the color-based Retinex assumption. Weiss used multiple observed images under varying lighting conditions [19].

Our method falls in the first category. Training paired patches are extracted from observed grayscale images and their shading components, where the mean (DC) of each patch is removed. Intrinsic image estimation is realized by estimating sparse feature vector at each pixel of the target image. To reconstruct the target image, we use MRF optimization: since dictionary atoms are DC-free, for the estimated patches centered at each pixel, we compute their optimal mean values by iterative conditional mode [5]. Final reconstruction of the target image is obtained by averaging DC returned estimated patches at overlapped pixels.

7.1 Experimental Results

We conducted experiments on the MIT intrinsic dataset [16], which contains 16 object images in realistic settings.

TABLE 1
Quantitative comparison of different methods.

Methods	Retinex	ExpertBoost	Our-local	Our-global
LMSE scores	0.041	0.040	0.033	0.031
Methods	Our-NN	Color Retinex	Weiss	
LMSE scores	0.030	0.030	0.021	

We used 15 objects in the dataset for training, and the left one for testing. For coupled dictionary learning, we extracted $50,000$ 8×8 patch pairs. The dictionary size was $K = 128$. The sparsity level was set as $t = 150$, which produced around 15 nonzero elements in the sparse vector of each patch. For space partitioning, we set $\zeta = 20$ and $\eta = 9$. For extraction of local patches from test images, denser sampling with more pixels overlap can give slightly better results, however, it also involves heavier computation. In practice we chose to sample 8×8 patches with 5 pixels overlap from each input image.

Figure 5 compares our results with those from Retinex [17] and ExpertBoost [11], both of which use a single grayscale image as input. For Retinex, the thresholding parameters were learned from the training data. For ExpertBoost, the parameter settings were the same as in [11]. Images in [16] can be generally classified into three levels of difficulty. The representatives are shown in Figure 5 where the difficulty levels go higher from "paper" to "frog". Our results of the "paper" and "raccoon" outperform those of Retinex and ExpertBoost. For both of them, some reflectance components contaminate the estimated shading images, while there is no such apparent contamination in our results. However, as Retinex and ExpertBoost do, our method also leaves cast shadows in the reflectance images. For the most difficult "frog", all methods fail to separate the shading/reflectance components, because surface changes caused by them are mixed at most of the pixels.

If memory is not a concern, for any test patch it is possible to search the closest sparse feature vector within its retrieved local cluster, and use the corresponding sparse feature vector in the target space as an estimate, instead of doing regression. By this means our method keeps efficiency, because we can still retrieve closest local cells efficiently, and NN search within local cells costs little. Figure 5-(g) shows that this means can generate even better results. For example, reflectance contamination is completely removed in the shading images of "paper" and "raccoon" objects.

Note that in intrinsic image estimation, one is only interested in recovering *relative* shading/reflectance images. To quantify the performance of different methods, we used the local mean squared error (LMSE) criteria proposed in [16]. Table 1 reports the LMSE scores of different methods, averaged over all 16 testings, where results of color Retinex [16] and Weiss's method using multiple input images [19] are from [16]. In Appendix E in the supplemental material, we also give example results for comparison of our method with color Retinex

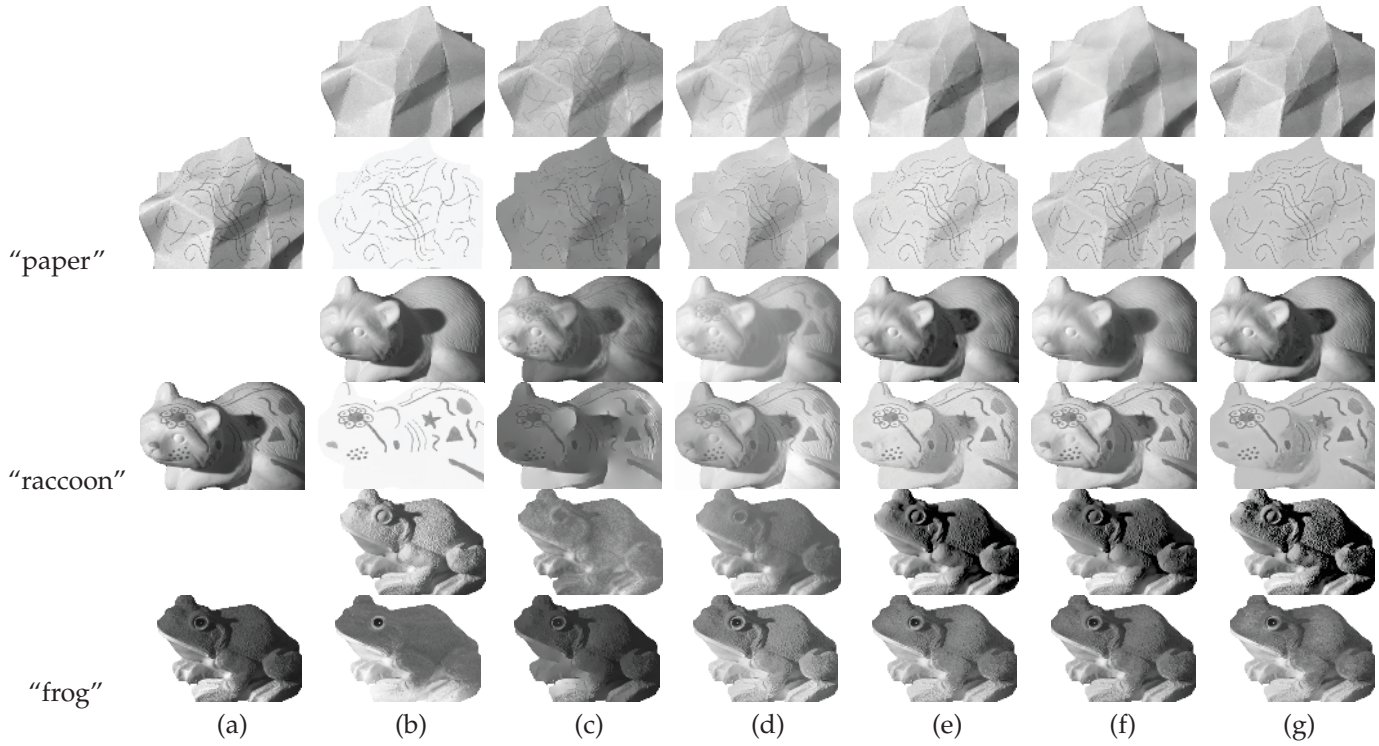


Fig. 5. Example results and comparison of intrinsic image estimation using different methods. In each column, for each pair of results, above is the shading component image, and below is the reflectance image. (a) Input observed images; (b) ground truth; (c) results of Retinex [17]; (d) results of ExpertBoost [11]; (e) our results just after local linear regression; (f) our results after global correction by histogram matching; (g) our results based on nearest neighbor searching within each retrieved local cluster.

and Weiss's method. Consistent with qualitative comparison, our method gives smaller quantitative error than other single grayscale image based methods, and as small as that using additional color information. Different values of t may have effects on our results. Interested readers may refer to Appendix F in the supplemental material for a discussion of sparsity level effect.

Benefits of Coupled Sparsity We have analyzed in Section 3.1.1 the advantages of coupled sparsity for image transformation over several alternative approaches, such as that based on independent sparse coding of training patch samples in source and target image spaces (dubbed *ScIndependent*), or directly estimating raw intensity values of target patches from sparse representations of source patches (dubbed *ScRaw*). We verified our analysis by experiments on intrinsic image estimation. The parameter settings of sparse coding and space partitioning in the source space were same for these three methods. We set $\zeta = 20$ and $\eta = 9$. The patch size was 8×8, $K = 128$, and the sparsity level was set as $t = 150$. The only difference between them is how they estimate the target patches. Table 2 reports averaged quantitative results over all 16 objects in the dataset [16], where we compare the two alternatives with our results right after local linear regression. Results of an example object are shown in Appendix H in the supplemental material. Except for using more regression functions and parameters, Appendix H shows that the estimated shading image by ScIndependent looks more blurry and with more albedo component remaining. Correspondingly more finer shape is left in its albedo image, which confirms what we have analyzed in Section 3.1.1. Upon using more regression functions, ScRaw gives similar results as ours. However, ScRaw is susceptible to corrupted training data, for example when training patch samples are contaminated by noise or missing data. To verify this, we performed another experiment by adding Gaussian noise to training images, with standard deviation set as 10. While our method can still give cleaner shading and albedo images, results from ScRaw are contamintated with noise. Table 2 quantitatively compares these methods, and interested readers can refer to Appendix H for visual quality comparison.

TABLE 2
Quantitative comparison with the alternative methods ScIndependent and ScRaw (see text for explanation).

Methods	ScIndependent	ScRaw	Our
LMSE scores	0.045	0.033	0.033
Methods		ScRaw-Noisy	Our-Noisy
LMSE scores		0.042	0.035

Comparison with the Related Coupled Sparse Coding Method For the concept of coupled sparse coding, we require sparse feature vectors of paired patches share the same support. A recent method for super-resolution [15] enforces corresponding low- and high-resolution image patches to share the same sparse feature values. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE 3
Quantitative comparison with the coupled sparse coding method in [15] (see text for explanation).

Methods	[15]-I	[15]-II	[15]-III	Our-II
LMSE scores	0.056	0.055	0.055	0.033

TABLE 4
LMSE scores of different methods with noisy inputs, corresponding to results in Appendix G.

Noise Std	0	2	4	6	8	10
Retinex	0.0108	0.0139	0.0192	0.0254	0.0311	0.0377
ExpertBoost	0.0146	0.0148	0.0156	0.0175	0.0200	0.0241
Our method	**0.0016**	**0.0038**	**0.0041**	**0.0046**	**0.0063**	**0.0070**

assumption seems intuitive for use in super-resolution. But it is not necessarily true for general types of image transformation. We conducted experiments on intrinsic image estimation to validate our claim. In particular, we replaced our coupled sparse coding and dictionary learning method with the method proposed in [15], and image transformation is still under the local parametric regression framework. For the method in [15], we used the same patch and dictionary sizes as in our method. To make the comparison clearer, we only used the basic procedure: results of both methods are directly from local linear regression. Table 3 reports averaged quantitative results over all 16 objects in the dataset [16], where results from the method in [15] under three different sparsity levels are presented. Sparsity levels I ([15]-I), II ([15]-II), and III ([15]-III) respectively correspond to 8, 15, and 30 nonzero elements in any sparse feature vector. Result by our method is under sparsity level II (Our-II). Qualitative comparison of an example object is shown in Appendix I in the supplemental material. Table 3 and Appendix I show that our method is good at separating shading/albedo components, and the method in [15] fails at such a general image transformation application.

Simultaneous Restoration and Transformation When images are to some extent corrupted, our method is relatively stable when computing their sparse feature vectors, thus inherently being more robust against corrupted data. We verified the claim by performing intrinsic image estimation from noisy inputs. In particular, we added Gaussian noise to input images, with standard deviation ranging from 0 to 10. Appendix G in the supplemental material compares our method with Retinex and ExpertBoost in these noisy settings, which shows that noise does influence the performance of different methods. Our method outperforms Retinex and ExpertBoost, and consistently gives cleaner results until standard deviation reaches 8. Quantitative comparison of different methods is reported in Table 4. Overall, our method can be considered as a simultaneous image restoration and transformation process.

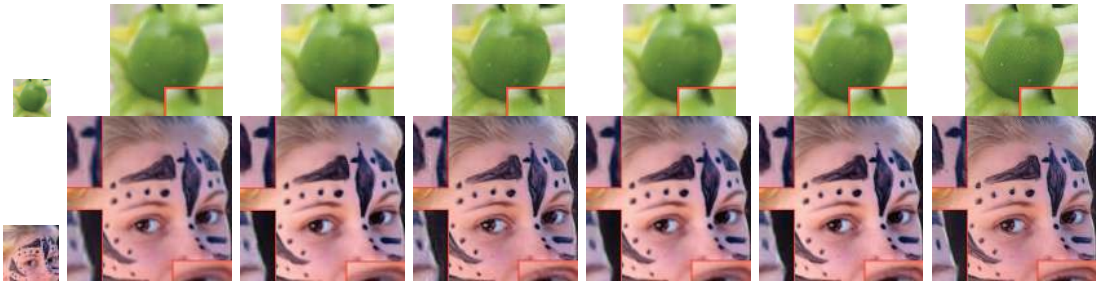
8 Image Super-Resolution

Super-resolution (SR) aims to generate a high-resolution (HR) image based on one or several given low-resolution (LR) images. The generated result should be sharp looking and smooth. In literature, there are many different methods for generic image SR, roughly categorized as reconstruction-based, interpolation with natural image priors [14], [13], and patch-wise learning based [4], [7], [15], [12], [50]. In general, the third category is better at super-resolving natural images with complex textures. Our proposed method falls into this category. We perform image SR in the learned sparse feature spaces. After estimating sparse feature vectors of densely overlapped patches, we essentially obtain multiple constraints for the desired HR image. To reconstruct the HR image, we make a minimum error boundary cut between adjacent patches [40]. Since the dictionary is DC free, for image SR the DC problem is easily treated by removing it from the LR input and returning back in the HR reconstruction.

8.1 Experimental Results

80 HR training images were collected from the Internet, and their LR counterparts were produced by first subsampling, and then upsampling via bicubic interpolation, from which $80,000$ LR and HR patch pairs were sampled to learn the coupled dictionaries. We used 5×5 patches for a 3x magnification factor, and 8×8 patches for a 4x magnification factor. The dictionary sizes were fixed as 4 times over-complete. The sparsity level was set as $t=110$. For space partitioning we set $\zeta=20$ and $\eta=9$, which can give $32,104$ local clusters out of $4,013,000$ training patch pairs. For extraction of local patches from test images, denser sampling with more pixels overlap can give slightly better results, however, it also involves heavier computation. In practice, we chose to sample 5×5 patches with 3 pixels overlap, or 8×8 patches with 5 pixels overlap, from each input image. For color images, we only applied our method to the illumination channel, and the chrominance channels were super-resolved by bicubic interpolation.

We compare our method on SR with closely related ones such as neighbor embedding [7], Yang's method [15] and soft edge prior [13]. Figure 6-(e) shows example results of our method just after local linear regression, and Figure 6-(f) shows the results after global correction by histogram matching. Columns (c) and (d) in Figure 6 are results from neighbor embedding and Yang's method. We used the same parameter settings as in [7] for column (c). The results of Yang's method in column (d) were produced using the code on their webpage. Both our method and [15] can generate sharper images than neighbor embedding. Compared with [15], our results have no spotty artifacts and halo effects around edges. We compare more of our results with neighbor embedding and soft edge prior [13] in Appendix J in the supplemental material. The result of neighbor embedding is not as sharp as ours. The soft edge prior can produce color attractive results, but also introduce some smoothing effect that is sometimes undesired. Compared with [13], our results look more photorealistic. To quantitatively compare different methods, we report in



(a) (b) (c) (d) (e) (f) (g)

Fig. 6. 3x super-resolution on girl and flower bud images (*better view the electronic version*). (a) LR input; (b) bicubic interpolation; (c) neighbor embedding [7]; (d) Yang's method [15]; (e) our results by local linear regression; (f) our final results; (g) ground truth.

TABLE 5
RMS errors of different super-resolution methods.

Images	Bicubic	[7]	[15]	[13]	Ours
girl	8.739	8.594	8.372	N/A	**7.919**
flower bud	4.938	4.859	4.681	N/A	**4.554**
head	9.515	9.368	N/A	9.3	**8.906**
Parthenon	11.4	11.73	10.8	10.7	**10.38**

Fig. 7. Super-resolution comparison with [21]. From left to right: bicubic interpolation, results from [21], our final results.

Table 5 the RMS errors for the images in Figure 6 and Appendix J. Consistent with the visual comparison, our method gives the lowest reconstruction errors using the original HR images as ground truth. Different choices of t may have effects on our results. Interested readers may refer to Appendix K in the supplemental material for a discussion of sparsity level effect.

Figure 7 compares our method with a single-image based SR method proposed by Glasner *et al.* [21]. Both methods can produce excellent results with sharp edges. Nevertheless, the method in [21] takes advantages of image self-similarities across scales and classical multi-image fusion based techniques, both of which may be only valid in the SR domain. In contrast, our method is proposed for more general types of image transformation, which include SR as an example, and also other applications such as intrinsic image estimation.

9 Conclusion and Future Work

In this paper, we propose a learning based framework for image transformation. Our framework is based on a local regression approach using sparse feature representations over learned dictionaries across image spaces. To learn the dictionaries, we propose the concept of coupled sparsity, and use an active set BCGD algorithm for coupled sparse coding. To learn mapping relations between image spaces, we perform parametric regression within small subsets of training image patch pairs. To this end, we propose a space partitioning scheme that can divide the high-dimensional but sparse feature spaces into easily retrievable local clusters. For any test image patch, our method can efficiently retrieve its closest local cluster and perform regression within the cluster. We applied our framework to intrinsic image estimation and super-resolution, and obtained the state-of-the-art performance. Our method is more robust to corrupted data, and can be considered as a simultaneous image restoration and transformation process.

In this work, coupled dictionary learning is performed in a purely unsupervised, data-driven fashion: dictionaries are learned for better reconstructing training patch pairs. It is possible to extend the current formulation to a supervised setting similar to [36], [35], i.e., dictionaries are learned to be better adapted to a specific image transformation task. We leave this extension in future research. In future we are also interested in extending our method on other image transformation applications.

References

[1] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin, Image analogies, *SIGGRAPH*, 2001.
[2] Z. Liu, Z. Zhang, and Y. Shan, Image-based surface detail transfer, *IEEE Computer Graphics and Applications*, vol.24, pp. 30-35, 2004.
[3] S. Bae, S. Paris, and F. Durand, Two-scale tone management for photographic look, *SIGGRAPH*, 2006.
[4] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael, Learning low-level vision, *IJCV*, 2000.
[5] J. Besag, On the statistical analysis of dirty pictures (with discussion), *J. Royal Stat. Soc., Series B*, Vol. 48, No. 3, pp. 259-302, 1986.
[6] S. Baker and T. Kanade, Limits on super-resolution and how to break them, *IEEE TPAMI*, 24(9), pp. 1167-1183, 2002.
[7] H. Chang, D.Y. Yeung, and Y. Xiong, Super-resolution through neighbor embedding, *CVPR*, 2004.
[8] C. Liu, H.Y. Shum, and W.T. Freeman, Face hallucination: theory and practice, *IJCV*, Vol. 75, No. 1, pp. 115-134, 2007.
[9] Y. Li and E.H. Adelson, Image mapping using local and global statistics, *Proc. of SPIE-IS&T Electronic Imaging*, SPIE Vol. 6806, 2008.
[10] D. Lin and X. Tang, Coupled space learning of image style transformation, *ICCV*, 2005.
[11] M.F. Tappen, E.H. Adelson, and W.T. Freeman, Estimating intrinsic component images using non-linear regression, *CVPR*, 2006.

[12] K. Kim and Y. Kwon, Single-image super-resolution using sparse regression and natural image prior, *IEEE Trans. on PAMI*, Vol. 32, No. 6, pp. 1127-1133, 2010.
[13] S. Dai, M. Han, W. Xu, Y. Wu, and Y. Gong, Soft edge smoothness prior for alpha channel super resolution, *CVPR*, 2007.
[14] J. Sun, Z. Xu, and H. Shum, Image super-resolution using gradient profile prior, *CVPR*, 2008.
[15] J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution via sparse representation, *IEEE Trans. on Image Processing*, 2009.
[16] R. Grosse, M.K. Johnson, E.H. Adelson, and W.T. Freeman, Ground-truth dataset and baseline evaluations for intrinsic image algorithms, *Proc. ICCV*, 2009.
[17] E.H. Land and J.J. McCann, Lightness and retinex theory, *Journal of the Optical Society of America*, 61(1):1-11, 1978.
[18] L. Shen, P. Tan, and S. Lin, Intrinsic image decomposition with non-local texture cues, *Proc. CVPR*, 2008.
[19] Y. Weiss, Deriving intrinsic images from image sequences, *Proc. ICCV*, Vol. 2, pp. 68-75, 2001.
[20] Y. Matsushita, S. Lin, S.B. Kang, and H.Y. Shum, Estimating intrinsic images from image sequences with biased illumination, *Proc. ECCV*, Vol. 2, pp. 274-286, 2004.
[21] D. Glasner, S. Bagon, and M. Irani, Super-resolution from a single image, *Proc. ICCV*, 2009.
[22] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching in fixed dimensions, *Journal of the ACM*, 45, 6, 1998.
[23] G. Shakhnarovich, T. Darrell, and P. Indyk, Nearest neighbor methods in learning and vision: theory and practice, *MIT Press*, 2006.
[24] Y. Weiss, A. Torralba, and R. Fergus, Spectral hashing, *NIPS*, 2008.
[25] M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE TIP*, 54(12), pp. 3736-3745, 2006.
[26] B.A. Olshausen and D.J. Field, Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37, pp. 3311-3325, 1997.
[27] S. Roth and M.J. Black, Fields of experts: A framework for learning image priors, *CVPR*, 2005.
[28] M. Yuan and Y. Lin, Model selection and estimation in regression with grouped variables, *J. Royal Stat. Soc. Series B*, 68:49-67, 2006.
[29] V. Roth and B. Fischer, The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms, *Proc. ICML*, 2008.
[30] P. Tseng and S. Yun, A coordinate gradient descent method for nonsmooth separable minimization, *Math. Program. Ser. B*, Vol. 117, pp. 387-423, 2009.
[31] K. Huang and S. Aviyente, Sparse representation for signal classification, *NIPS*, 19, pp. 609-616, 2007.
[32] M. Aharon, M. Elad, and A.M. Bruckstein, The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations, *IEEE Trans. SP*, 54(11), 2006.
[33] H. Lee, A. Battle, R. Raina, and A.Y. Ng, Efficient sparse coding algorithms, *NIPS*, 2007.
[34] J. Wright, A.Y. Yang, A. Ganesh, S. Sastry, and Y. Ma, Robust face recognition via sparse representation, *IEEE TPAMI*, 2008.
[35] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, Supervised Dictionary Learning, *NIPS*, 2008.
[36] J. Mairal, F. Bach, and J. Ponce, Task-Driven Dictionary Learning, to appear in *PAMI*, 2011.
[37] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, Learning invariant features through topographic filter maps, *CVPR*, 2009.
[38] J. Sivic and A. Zisserman, Video Google: A text retrieval approach to object matching in videos, *ICCV*, pp. 1470-1477, 2003.
[39] H. Jegou, M. Douze, and C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, *ECCV*, 2008.
[40] A.A. Efros and W.T. Freeman, Quilting for texture synthesis and transfer, *ACM Conf. on Comp. Graphics and Interactive Tech.*, 2001.
[41] M. Osborne, B. Presnell, and B. Turlach, A new approach to variable selection in least squares problems, *IMA J. Numer. Anal.*, Vol. 20, pp. 389-403, 2000.
[42] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, Least angle regression, *Ann. Stat.*, 32(2), 2004.
[43] R. Tibshirani, Regression shrinkge and selection via the Lasso, *Journal of Royal Statist*, Soc B., Vol. 58, No. 1, pp. 267-288, 1996.
[44] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, Convex optimization with sparsity-inducing norms, in Optimization for Machine Learning, *MIT Press*, 2011.
[45] R. Gray and D. Neuhoff, Quantization, *IEEE Trans. on Inf. Theory*, 44(6), pp. 2325-2383, 1998.
[46] G.J. McLachlan and K.E. Basford, Mixture Models: Inference and Applications to Clustering, *New York: Dekker*, 1988.
[47] G. Yu, G. Sapiro, and S. Mallat, Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity, *arXiv:1006.3056*, 2010.
[48] D. Nettleton, A. Orriols-Puig, and A. Fornells, A study of the effect of different types of noise on the precision of supervised learning techniques, *Artif. Intell. Rev.*, 2010.
[49] X. Wang and X. Tang, Hallucinating face by eigentransformation, *IEEE Trans. on SMC - Part C*, 2005.
[50] Q. Wang, X. Tang, H. Shum, Patch based blind image super resolution, *ICCV*, 2005.
[51] X. Tang and X. Wang, Face sketch recognition, *IEEE TCSVT*, 2004.
[52] X. Wang and X. Tang, Face photo-sketch synthesis and recognition, *IEEE Trans. on PAMI*, 2008.
[53] D. Nister and H. Stewenius, Scalable recognition with a vocabulary tree, *CVPR*, 2006.
[54] S. Lloyd, Least squares quantization in PCM. *IEEE Trans. on Inf. Theory*, 28(2), pp. 129-137, 1982.

Kui Jia received the B.Eng. degree in electrical engineering from Northwestern Polytechnic University, China, in 2001, the M.Eng. degree in electrical and computer engineering from the National University of Singapore in 2003, and the Ph.D. degree in computer science from Queen Mary, University of London, London, U.K., in 2007. He is currently a Research Scientist with the Advanced Digital Sciences Center, University of Illinois at Urbana-Champaign. His research interests include computer vision, machine learning, and image processing.

Xiaogang Wang received the BS degree from the University of Science and Technology of China in 2001, the MS degree from the Chinese University of Hong Kong in 2003, and the PhD degree from the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology in 2009. He is currently an Assistant Professor in the Department of Electronic Engineering at The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Xiaoou Tang received the BS degree from the University of Science and Technology of China, Hefei, in 1990, the MS degree from the University of Rochester, New York, in 1991, and the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a Professor in the Department of Information Engineering and Associate Dean (Research) on the Faculty of Engineering of the Chinese University of Hong Kong. He worked as the group manager of the Visual Computing Group at Microsoft Research Asia from 2005 to 2008. He received the Best Paper Award from the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He was a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009. His research interests include computer vision, pattern recognition, and video processing. He is a fellow of the IEEE.