Blind Separation of Superimposed Moving Images Using Image Statistics

Kun Gai, Zhenwei Shi, and Changshui Zhang

Abstract—We address the problem of blind separation of multiple source layers from linear mixtures thereof, involving unknown linear mixing coefficients and unknown motions of layers in each mixture. Such mixtures can be caused in photography by the presence of a transparent medium, like a window glass, when the camera or the medium moves between snapshots. To understand how to achieve correct separation, we study the statistics of natural images in the Labelme data set. We not only confirm the well-known sparsity of image gradients, but also discover new joint behavior patterns of image gradients. Based on these statistical properties, we develop a sparse blind separation algorithm to estimate both layer motions and linear mixing coefficients and then recover all layers. This method can handle general parameterized motions, including translations, scalings, rotations and other transformations. In addition, the number of layers is automatically identified, and all layers can be recovered even in the under-determined case where mixtures are fewer than layers. The effectiveness of this technology is shown in both simulated and real superimposed images.

Index Terms—Blind source separation (BSS), reflection, transparency, motion, image statistics.

1 INTRODUCTION

When photographing through a transparent surface, like a window glass, we often obtain additive mixtures of two layers: one layer is the transmitted scene behind the surface and the other is the reflected scene in front of the surface. Such kind of mixture images can seriously disturb human perception, as well as many computer vision algorithms, such as segmentation, tracking and object detection. Thus, the need of separating mixtures and recovering the component layers arises.

When only one mixture image is available, the separation is massively ill-posed. (although Levin et al. attempted it in [1] and [2] and then two-layer user-assisted separation systems are developed [3], [4], the systems are not automatic). However, when two or more mixtures are available, automatic separation can be achieved by exploiting the diversity of different mixtures. From one mixture to another, some layer properties probably change. First, layers may have relative motions because of movements of the camera, the glass surface or the target object. Such movements may be inevitable (e.g., due to hand jitter), and can also be introduced deliberately for separation. Second, layers’ overall intensity may vary, since the changing reflection angles along with the movements will alter the distributions between reflected and transmitted lights. The different intensity can also be caused by different camera settings, lighting conditions, or by the introduction of polarization filters as in [5], [6]. Such diversity will lead to some changes of layers’ mixing coefficients. Our goal is to estimate the diversities of layer motions and mixing coefficients between mixtures and then recover all layers.

1.1 Previous works on the separation problem

The separation of mixture images is in general very challenging, and received much attention in past years. This problem naturally belongs to the scope of Blind Source Separation (BSS) [7], and a number of BSS-based approaches have been proposed. Some methods introduce polarization filters in photography to alter the mixing coefficients of layers, and perform Independent Component Analysis (ICA) [5], [6], [8], [9], [10]. Some other methods obtain different mixtures with different focuses, and use multi-channel Blind Deconvolution (BD) [11], [12]. These traditional BSS methods, including ICA and BD, assume static mixing, which is not always easy to be satisfied in practice.

Another direction for the separation is to utilize relative motions of layers. Various technologies are proposed to extract multiple motions from image sequences, e.g., [13], [14], [15], [16], [17], [18], [19]. They focus on only motion recovery, whereas the layer restoration is not considered. Based on motions output by these motion estimation methods, a min/max alternative method [20] is then developed to separate two-layer mixtures from layers’ parameterized motions (and it is extended for stereo matching in [21]). It needs a large amount of images with motions of various directions. When there are only two or several mixtures, it has the degeneracy problem [20], which causes unwanted stripes in result layers. Moreover, it assumes fixed mixing coefficients, which can be easily violated by the changes of reflection angles or camera settings.

Recently, the BSS framework is extended to handle spatial shifts of layers in addition to unknown mixing
coefficients. Be'ery and Yeredor propose 2D-AC-DC [22], [23] to separate superimposed images of two shifting layers. It has the local minimal problem when estimating layer shifts. In our previous work [24], we developed a fast algorithm named Sparse Blind Separation with Spatial Shifts (SPBSS) to handle such mixtures of multiple layers, without any local minimal problem in the estimation of layer shifts. Nevertheless, both SPBSS and 2D-AC-DC assume motions to be only uniform translations, and hence are still limited for real applications.

1.2 Our contributions and paper organization

In this paper, we extend the BSS framework to handle mixtures with not only unknown mixing coefficients but also unknown parameterized layer motions (including translations, scalings, rotations and other transformations), in addition to an unknown layer number. Such type of mixtures are more general and more applicable than the ones used in previous works. To the best of our knowledge our proposed approach is the only work that can handle such mixtures (some preliminary results of this work can also be found in our conference paper [25]). In addition, our method has a number of desirable properties. (1) It can deal with an arbitrary number of layers from only two mixtures. (2) The layer number is automatically identified. (3) Even when the layer number is wrongly set by users, layers still can be well reconstructed. (4) Our method has no degeneracy problem and the recovered layers are of high image quality.

Besides the algorithm, this paper also provides image statistics based on more than 130,000 natural images in Labelme dataset [26]. We confirm the well known sparsity property of image gradients, and discover new joint behavior patterns of image gradients. These statistics help us better understand natural images and simplify the blind separation problem. We believe that other computer vision tasks can also benefit from them.

The rest of this paper is organized as follows. Section 2 gives the mixing model of moving superimposed images. To find some clues for the separation, we study the statistics of natural images in Section 3. By use of image statistics, we present a new blind separation algorithm to estimate motions and mixing coefficients of layers in Section 4. Subsequently, the identification of the layer number is discussed in Section 5. Then Section 6 presents a novel layer-reconstruction approach. Section 7 and 8 show the experiments on simulated mixtures and real superimposed images, respectively. Finally, we close with a conclusion.

2 Problem formulation

As discussed in Section 1, in each mixture, the layers have unknown and possibly different motions in addition to unknown and possibly different mixing coefficients. Here we assume that mixtures are linear w.r.t. layers, as in many other works (e.g., [3], [10], [20], [23], [24]), and formulate the moving mixing model of $m$ mixtures with $n$ layers as

$$I_i(x) = \sum_{j=1}^{n} a_{ij} L_j(f_{ij}(x)), \quad i = 1, \cdots, m,$$  \hspace{1cm} (1)

where $I_i$ is the $i$th mixture, $L_j$ is the $j$th layer, and $x = (x_1,x_2)^\top$ is a 2D integer vector that represents the pixel location. The mixing coefficient and the motion transformation are described by $a_{ij}$ and $f_{ij}()$, respectively, and they are called mixing parameters (here we use parametric motions and so $f_{ij}()$ is indeed controlled by a set of motion parameters). Without loss of generality, we take the component layers in the first mixture as reference layers, and thereby $a_{ij} = 1$ and $f_{ij}(x) = x$. For simplicity, we use a matrix $A$ to describe all mixing coefficients, and $A = [a_{ij}] = [A_1^\top, \cdots, A_n^\top]^\top$, where $A_i$ denotes the $i$th row of $A$.

Here only the observed mixtures $I_1, \ldots, I_m$ are known. The underlying layers $L_1, \ldots, L_n$, the layer number $n$, and the mixing parameters $a_{ij}$ and $f_{ij}$ are all unknown. Our goal is to first estimate the layer number $n$ and the mixing parameters, and then recover the underlying layers $L_1, \ldots, L_n$. Such separation problem is massively ill-posed. For example, one can recognize each mixture itself as a different layer without performing any separation and the mixing model can still be perfectly satisfied. To correctly identify true underlying mixing parameters and layers, we need to introduce some suitable hypotheses, like the independence of layers in ICA and BD. Unlike the models in ICA and BD, the moving mixing model (1) is nonlinear w.r.t. mixing parameters due to the presence of layer motions. The nonlinearity makes the problem much more complicated.

Figure 1 shows a demonstration of three moving mixing images and their gradients. Column 1 shows 3 original layers, and Column 2 shows 3 mixtures of them. The layer motions are planar transformations, which can be represented by the following 8-parameters form:

$$\text{planar}(x) = Pe^\text{ext}(x + t_p),$$ \hspace{1cm} (2)

where

$$e^\text{ext}(x) = [x_1, x_2, x_1^2, x_1x_2, x_2^2]^\top,$$ \hspace{1cm} (3)

$$P = \begin{bmatrix} p_3 & p_4 & p_7 & p_8 & 0 \\ -p_4 + p_5 & p_3 + p_6 & 0 & p_7 & p_8 \end{bmatrix},$$ \hspace{1cm} (4)

and

$$t_p = [p_1, p_2]^\top.$$ \hspace{1cm} (5)

Parameters $p_1$ and $p_2$ denote translations, $p_3$ denotes scalings, $p_4$ denotes rotations (together with $p_3$), and $p_5$
to \( p_8 \) denote some other warping effects. We assemble 8 planar parameters in \( f_{ij}(x) \) into a vector, and use \( p_{ij} \) to denote this vector. The mixing parameters in the 3rd mixture are as follows.

\[
A_3 = [1.00, 0.625, 1.75], \\
p_{31}^{31} = [-3, -4, 1, -0.06, 0, -0.01, 10^{-4}, 10^{-1}], \\
p_{32}^{32} = p_{33}^{33} = [1, 3, 0.996, 0.06 -0.02, 0, 0, 0].
\]

Although the 2nd and the 3rd layers have identical motions, they can still be discriminated owing to different mixing coefficients. Column 3 of Fig. 1 shows image gradients (\( L_2 \)-norms) of the mixtures. The gradients are sparse, as most pixels are black. From these observations, we can clearly distinguish a certain amount of edges for every underlying layer. It inspires us to make use of image gradients for the separation. Column 4 shows the separation results, by the proposed method in this paper.

### 3 Statistics of Natural Images

In this section, we explore statistical properties of natural images from a large scale image set, expecting to find some clues for the correct separation. The used image set is Labelme [26], which contains more than 130,000 pictures and almost all common image categories, e.g., indoor, outdoor, city, animals and sea scenes, and thus is suitable to show statistical properties of natural images.

The original pixels of images are integers between 0 and 255. The gradient filter used in this paper, whose outputs should be 2-D vectors, is a combination of vertical and horizontal \([-1, 1]\) filters. Thus, the image gradients are also integers, which are between \(-255\) and 255. For these integral variables, we obtain the probability density function (PDF) of each of them and the joint PDF of any pair of them which are at a certain distance by counting the occurring frequency of their every possible value in the whole data set. From these PDFs we confirm the sparsity of gradients and add new findings about joint behaviors of gradients.

#### 3.1 Sparsity

Figures 2(a) and 2(b) show PDFs of original pixels and image gradients, respectively. Although original pixels are not sparse, image gradients are very sparse. The sparsity denotes that most gradients are approximately equal to zero and only a small part of them are significantly different from zero. The gradients significantly different from zero are called significant gradients, while others are called nonsignificant ones. The sparsity of gradients has been shown in previous works, e.g., [1], [10], [27], based on several or dozens of images belonging to one or several certain categories. Here we give a stronger validation by a much larger dataset containing almost all common image categories.

To check how sparse image gradients are, we use the exponential power (also known as generalized Gaussian) density as references, which has a form of

\[
p(y) = 0.5b\lambda^{1/b}\Gamma^{-1}(1/b)e^{-\lambda|y|^b},
\]

(9)

where \( \lambda \) and the exponent power \( b \) control the variance and the sparsity, respectively, and \( \Gamma(\cdot) \) is the Gamma function. The smaller \( b \) is, the more sparse the density is. When \( b = 2 \) and \( b = 1 \), the density (9) becomes the Gaussian and the Laplacian densities, respectively. The logarithmic density of (9) has an exponential form, as

\[
\log(p(y)) = \theta - \lambda|y|^b,
\]

(10)

where \( \theta \) is a parameter related to \( \lambda \) and \( b \). When \( b > 1 \), (10) is concave w.r.t. \( |y| \). When \( b \leq 1 \), (10) is convex w.r.t. \( |y| \) and we say the density (9) is a sparse density. Figure 2(c) uses dashed lines to show logarithmic densities (10) with \( b = 2, 1, 0.5, 0.4 \) and 0.3, and uses a solid line to show the logarithmic density of image gradients. We adjust \( \theta \) and \( \lambda \) to make dashed lines meet the solid line on \( |y| = 0 \) and 184. The solid line for image gradients is convex and is between the dashed lines with \( b = 0.4 \) and \( b = 0.3 \) for most \( |y| \), showing the strong sparsity of image gradients.

#### 3.2 Noncorrelation

Despite of the well known sparsity, the relationships of different gradients have been little explored. In this subsection, we check the linear dependence of different gradients. Figure 3(a) and 3(b) show the intra-image correlations of pixels and gradients, respectively. The latter is calculated by the following equation:

\[
cor(z) = \sum_{x} \sum_{k_1,k_2} \nabla_{k_1} \text{Pic}_i(x) \nabla_{k_2} \text{Pic}_i(x + z),
\]

(11)

where \( \text{Pic}_i \) denotes the \( i \)th picture, \( \nabla(\cdot) \) denotes the gradient operator, and \( k_1 \) and \( k_2 \) are the subscripts of
3.3 Joint behaviors and conditioned sparsity

We continue to study joint behaviors of different gradients in a same image by their joint PDFs. Four joint PDFs of vertical gradient pairs at \( L_\infty \) distances equal to 1, 2, 3 and 5 are illustrated in the four leftmost images in the top row of Fig. 4 (For horizontal gradient pairs we find almost the same joint PDF images and the same properties and thus their joint PDF images are not plotted). The plotted joint PDF images show interesting structures: there are relatively bright lines along two axes (called axis lines) and relatively bright lines along bisectors of axes (called bisector lines), and the bisector lines become weak rapidly (from the leftmost image to right images) when the distance of the gradient pair increases. The bright axis lines indicate that there is a relatively high probability that one gradient in the pair is zero, and the bright bisector lines indicate that there is a relatively high probability that two gradients have the same absolute value. Such joint behavior patterns seem surprising: wavelet coefficients of natural images based on Gabor filters have been reported to be also sparse and approximately uncorrelated, but such behavior patterns are not yet found in them [28]. The reason may be explained by the following example.

Consider a window containing two smooth areas, and assume that all pixels in the window are equal to either \( val_A \) for one area or \( val_B \) for the other area. Whatever shapes of two areas are, a vertical or horizontal gradient in the window, being the difference of two adjacent pixel values, either is zero (when not on the edge) or has the same absolute value \( |val_A - val_B| \), leading to the found behavior patterns. Other filters like Gabor filters have complex dependences on \( 2 \times 2 \) or larger patches of pixels, and will not result in the same behavior patterns.

From the four leftmost joint PDF images in Fig. 4 we can find that the bisectors lines are as bright as the axis lines only when the distance of the gradient pair is 1. When the distances are larger than 1, the bisector lines are relatively weak compared to the axis lines. Since the shown joint PDF images are actually over-saturated in axis lines, to better show relative differences between probabilities in axis lines and bisector lines, we plot corresponding conditioned PDFs of gradient pairs in the bottom row of Fig. 4. These conditioned PDF images show a conditioned sparsity property: if the distance of a gradient pair is larger than a threshold value, which we denote by \( dis_{stat} = 1 \), whatever the value of one gradient is, the probabilities of the other gradient conditioned on this gradient mainly concentrate on zero. This conclusion also holds for gradient pairs of different directions, even without a threshold distance. For example, the rightmost images in the top row and in the bottom row of Fig. 4 illustrate the joint PDF and the conditioned PDF of a gradient pair of different directions at a same location, respectively, which also show the conditioned sparsity.

3.4 Independence between images

We now come to the relationship between different images. We randomly sample 1 million image pairs from Labelme, and in every image pair we randomly sample 1 million location pairs, one from the first image and the other from the second image. By counting the occurring frequency of every possible joint value of pixel pairs and gradient pairs on our sampled location pairs, we obtain joint PDFs of pixel pairs and of gradient pairs from different images. Based on the joint PDFs, we calculate the normalized mutual information (NMI). The NMI of pixel pairs from different images is only \( 4.6 \times 10^{-7} \), and the NMIs of vertical, horizontal, and different directional gradient pairs from different images are only \( 1.4 \times 10^{-7} \), \( 1.3 \times 10^{-7} \), and \( 1.3 \times 10^{-7} \), respectively. It shows that gradients from different images as well as pixels from different images are independent of each other.

4 Sparse blind separation with motions

Based on image statistics, we propose an algorithm named sparse blind separation with motions (SPBS-M) to estimate all mixing parameters from the moving superimposed images. In this section we assume the mixture number is 2, and the layer number is known, equal to \( n \). To make different layers discriminable, it is required that for any two layers there are some diversities either in layer motions or in mixing coefficients (or in both). Besides, to guarantee the accuracy of distinguishing different parameters, we further require that for any two layers their motions are either identical or different enough: \( \forall j_1, j_2 \), if \( j_1 \neq j_2 \), either \( f_{2j_1}(x) = f_{2j_2}(x) \), or for almost all \( x \), \( \|f_{2j_1}(x) - f_{2j_2}(x)\|_\infty > dis_{stat} = 1 \). This can guarantee that when images are down sampled by a scaling factor 2, different layer motions still can be distinguished on most locations. The extension of our algorithm to more mixtures and the identification of the layer number will be discussed in Subsection 4.7 and Section 5, respectively.
4.1 Hypotheses

For convenience, we use $\nabla \imath(u(x))$ and $D(\imath(u(x)))$ to denote warped gradients of image $\imath(x)$ by motion $u(x)$ and derivatives of $\imath(u(x))$ w.r.t. $x$, respectively, i.e.,

$$\nabla \imath(u(x)) = \left. \frac{d\imath(y)}{dy} \right|_{y=u(x)}, \quad (12)$$

$$D(\imath(u(x))) = \frac{d\imath(u(x))}{dx} = \frac{d\imath}{dx}(x) \nabla \imath(u(x)), \quad (13)$$

and assume that the layer gradient $\nabla_k L_i(x)$ ($1 \leq i \leq n$), where $k$ ($k=1$ or $2$) denotes the $k$th element of a gradient vector, is a zero-mean random process w.r.t. $L_i$. Based on image statistics, we give the following hypotheses:

1) Conditioned sparsity: for any $k_1$, $k_2$ and $i$, when $|x-y|_{\infty} > dis_{stat} = 1$, whatever the value of $\nabla_{k_2} L_i(y)$ is, $\nabla_{k_1} L_i(x)$ is sparse conditioned on $\nabla_{k_2} L_i(y)$.

2) Noncorrelation: for any $k_1$, $k_2$ and $i$, when $x \neq y$, $\nabla_{k_1} L_i(x)$ and $\nabla_{k_2} L_i(y)$ are uncorrelated.

3) Independence between layers: for any $k_1$, $k_2$, $x$ and $y$, when $i \neq j$, $\nabla_{k_1} L_i(x)$ and $\nabla_{k_2} L_j(y)$ are independent.

Note that the conditioned sparsity says that (when $|x-y|_{\infty} > dis_{stat} = 1$) the probability of $\nabla_{k_1} L_i(x)$ being near zero conditioned on $\nabla_{k_2} L_i(y)$ is high for any value of $\nabla_{k_2} L_i(y)$, and thus it also guarantees the marginal sparsity: for any $i$ and $k_1$, the probability of $\nabla_{k_1} L_i(x)$ being near zero is high, i.e., $\nabla_{k_1} L_i(x)$ is sparse.

4.2 Motion objective function

Now consider the estimation of layer motions. Traditional mixture motion (or layered motion) technologies, e.g., [13], [14], [15], [17], [19], assume that layers’ mixing coefficients are fixed to be 1, which can be easily violated by changes of reflection angles or lighting conditions, or the introduction of polarization filters, and thus are not suitable here. Even in the case where mixing coefficients are fixed, different researchers have found that results of the mixture motion methods are not reliable for superimposed images [8], [20]. The mixture motion methods assume that spatiotemporal derivatives of observed images in a local patch only come from a single layer so that layer motions can be extracted from these derivatives using mixture technologies like clustering.

It is true that there is a high probability that spatial derivatives (i.e., gradients) of mixtures are contributed by only one layer due to the sparsity and independence between layers. However, the practically used temporal derivatives of mixtures are usually contributed by all layers. Take the temporal difference filter as the used temporal derivative filter for example. The temporal derivative of $I_j$ is gotten by the temporal difference $I_j(x) - I_1(x) = \sum_j (L_j(f_j(x)) - L_j(x))$. For the location where $\nabla L_i(x)$ is significant (so that $\nabla I_1(x) = \nabla L_i(x)$ with a high probability), the temporal difference of mixtures $I_j(x) - I_1(x)$ is equal to the first layer’s temporal difference $(L_i(f_j(x)) - L_i(x))$ only when the temporal differences of other layers $(L_j(f_j(x)) - L_j(x), j \neq 1)$ are zero. This condition can be seriously violated when $L_j(x)$ and $L_j(f_j(x))$ are not in the same level and smooth area ($j \neq 1$) (i.e., for one of other layers, motion flow in location $x$ has moved out of its located level area), and may be unsatisfied everywhere if one layer motion $f_j(x)$ is large or level and smooth in one layer are small (e.g., due to some textures) ($j \neq 1$). Therefore the gotten motions are often inaccurate. Using other types of temporal derivative filters, e.g., [29], also can not solve the problem. Below we show how to reliably and accurately find layer motions from superimposed images in a different manner, and in a more general case where mixing coefficients are unknown and probably different in different mixtures.

To search for correct layer motions, we move mixture $I_1$ with a searching motion $u(\cdot)$. Consider the gradient correlation between warped $I_1$ and $I_2$ on location $x$, as

$$o(u, x) = \mathbb{E} \left[ \langle D(I_1(u(x))), \nabla I_2(x) \rangle \right], \quad (14)$$

where $\langle \cdot, \cdot \rangle$ represents the inner product of two vectors. By use of the mixing model (1), the expected correlation (14) can be expanded as:

$$o(u, x) = \sum_j a_{2j} \mathbb{E} \left[ \nabla I_1(u(x)) \frac{du(x)}{dx} \frac{df_{2j}(x)}{dx} \nabla L_j(f_{2j}(x)) \right] + \sum_{i \neq j} a_{2j} \mathbb{E} \left[ \nabla I_1(u(x)) \frac{du(x)}{dx} \frac{df_{2j}(x)}{dx} \nabla L_j(f_{2j}(x)) \right]. \quad (15)$$

Due to the independence between layers, the second term in the right of the above equation is equal to

$$\sum_{i \neq j} a_{2j} \mathbb{E} \left[ \nabla I_1(u(x)) \frac{du(x)}{dx} \frac{df_{2j}(x)}{dx} \mathbb{E} \left[ \nabla L_j(f_{2j}(x)) \right] \right] = 0. \quad (16)$$

Consequently, $o(u, x)$ is a weighted sum of each layer’s gradient correlation, as

$$o(u, x) = \sum_j a_{2j} \phi_j(u, x), \quad (17)$$

where

$$\phi_j(u, x) = \mathbb{E} \left[ \nabla L_j(u(x)) \frac{du(x)}{dx} \frac{df_{2j}(x)}{dx} \nabla L_j(f_{2j}(x)) \right]. \quad (18)$$

Intuitively, if each layer’s gradient correlation $\phi_j(\cdot, x)$ (for a given $x$), which peaks at the correct layer motion $f_{2j}$, has a sharp shape, $o(\cdot, x)$ will also peak at every correct layer motion $f_{2j}$. This is actually guaranteed by the intra-layer noncorrelation. On one hand, when $u(\cdot) \neq f_{2j}(\cdot), \nabla L_j(u(x))$ and $\nabla L_j(f_{2j}(x))$ are uncorrelated. Then we have

$$\phi_j(u, x) = \mathbb{E} \left[ \nabla L_j(u(x)) \frac{du(x)}{dx} \frac{df_{2j}(x)}{dx} \mathbb{E} \left[ \nabla L_j(f_{2j}(x)) \right] \right] = 0. \quad (19)$$

On the other hand, when $u(\cdot) = f_{2j}(\cdot)$, we get

$$\phi_j(f_{2j}, x) = \mathbb{E} \left[ \nabla L_j(f_{2j}(x)) \frac{df_{2j}(x)}{dx} \frac{df_{2j, j}(x)}{dx} \nabla L_j(f_{2j}(x)) \right]. \quad (20)$$

Note the term in the expectation is positive determined, being always larger than or equal to zero. This term is not always equal to zero since $\nabla L_j(x)$ is not a zero random variable. Therefore the expected value $\phi_j(f_{2j}, x)$ must be larger than zero. In all, we get $\phi_j(u, x) = 0$ when $u \neq f_{2j}$, and $\phi_j(u, x) > 0$ when $u = f_{2j}$.
The searching of \( p_1 \) and \( p_2 \) is not considered here, and will be left to the bottom level. The selection order is that we first search for \( p_3 \) and \( p_4 \), then for \( p_5 \) and \( p_6 \), finally for \( p_7 \) and \( p_8 \). Such three-times-searches process is done iteratively until the objective function does not increase. In the middle level, we use the hierarchical brute force to search for the two selected parameters. With a specified initial searching interval, we test all discrete values of the selected parameters to find the optimal one. In each test we use the bottom level searching to match the optimal translation. After the first brute force search, we halve the interval and test the optimal solution and its 8 new neighbors to refine the result. Such refining process is done by \( T \) times to achieve a satisfactory precision. In the bottom level, we need to match the optimal translation \( t_p = [p_1, p_2] \) with given \( p_3 \) to \( p_8 \). In such case the motion objective function (23) is actually a correlation function w.r.t. \( t_p \). Thus the search can be efficiently done through the Fast Fourier Transform (FFT), with a complexity of \( O(N \log(N)) \).

The basic idea of such alternative scheme can be explained intuitively. First, we use scalings and rotations as well as translations to match a certain amount of significant gradients (edges) of a layer. Then, all planar parameters are alternatively refined to match more and more significant gradients of this layer. In each iteration, the objective function will not decrease, and so the convergence is guaranteed. Suppose the initial searching interval is set to give \( K \) discrete levels for each parameter in \( p_3 \) to \( p_8 \). The complexity of each iteration in the top level alternative optimization is \( O(3(K^2 + 8T)N \log(N)) \). In experiments, we set \( K \) and \( T \) to be no more than 20 and 5, respectively. The initial searching intervals for the first order parameters \( p_3 \) to \( p_6 \) and for the second order parameters \( p_7 \) and \( p_8 \) are set to be about 0.02 and 10\(^{-4} \), respectively. The optimization scheme usually achieves the convergence within only several iterations.

Directly maximizing the motion objective function only gives one layer motion at the highest peak. Other mixing parameters are still unknown. Motivated by sparse ICA [10], we study the scatter plot methods in the moving mixing case, and present a novel joint clustering formulation to estimate all mixing parameters.

### 4.3 Scatter plots and feature lines

Given a motion \( u \) found by maximizing the motion objective function, we draw 2D derivative points \( (D_k(I_1(u(x))), \nabla_k I_2(x)) \) for all \( k \) and \( x \) in a plane, which is called the scatter plot with motion \( u \), and consider what will happen in this scatter plot.

By use of the mixing model (1), we can expand elements in the 2D derivative points as

\[
\begin{align*}
    D_k(I_1(u(x))) &= \sum_{i \neq j} \frac{\partial u_i(x)}{\partial x_k} \nabla L_i(u(x)) + \frac{\partial u_i(x)}{\partial x_k} \nabla L_j(u(x)), \\
    \nabla_k I_2(x) &= \sum_{i \neq j} \frac{\partial f_{ij}(x)}{\partial x_k} \nabla L_i(f_{2i}(x)) + a_{ij} \frac{\partial f_{ij}(x)}{\partial x_k} \nabla L_j(f_{2j}(x)),
\end{align*}
\]
the points in the scatter plot into the radian space by axes because of the unmatched layers. If we map all

On one hand, if

where two terms respecting the \( j \)th layer are listed separately for analysis. Due to the independence-between-layers and sparsity, for any location \( x \) where \( \nabla L_j(f_2(x)) \) is significant, there is a high probability that the gradients of other layers are nonsignificant, i.e., \( \nabla L_i(g(x)) = 0 \) for any \( i (i \neq j) \) and \( g \). It means: for most \( x \in B_j = \{ x | \nabla L_j(u(x)) \) or \( \nabla L_j(f_2(x)) \) is significant\},

On one hand, if \( u \) matches this layer, i.e., \( u = f_2, \) for most \( x \in B_j \)

Thus, corresponding 2D points \( (D_k(I_1(u(x))), \nabla_k I_2(x)) \) are on a line with a slope equal to the corresponding mixing coefficient \( a_{2j} \). On the other hand, when \( u \) does not match this layer, i.e., \( u \) and \( f_2 \) is different, for most \( x \in B_j \) we have \( |u(x) - f_2(x)|_\infty > \text{dis}_{\text{stat}} = 1 \) as \( u \) is identical with one layer motion and different layer motions are assumed to be different enough. Then the conditioned sparsity tells that \( \nabla L_j(u(x)) \) and \( \nabla L_j(f_2(x)) \) are sparse conditioned on each other, and hence most significant values of \( \nabla L_j(u(x)) \) and \( \nabla L_j(f_2(x)) \) do not appear simultaneously. Thereby, corresponding 2D points \( (D_k(I_1(u(x))), \nabla_k I_2(x)) \) are on two axes.

In summary, if there exist \( g \) \((0 \leq g \leq n)\) layers matched by motion \( u \), in the scatter plot with \( u \) there will be \( g \) clusters along \( g \) lines with slopes equal to the corresponding mixing coefficients. Such lines are called feature lines. Besides, there are 2 additional clusters on axes because of the unmatched layers. If we map all the points in the scatter plot into the radian space by \( \arctan \left( \frac{\nabla_k I_2(x)/D_k(I_1(u(x)))}{D_k(I_1(u(x)))} \right) \), then in the corresponding radian density plot there will be \( g + 3 \) peaks (Clusters along axes will cause 3 peaks at \( -\frac{\pi}{2}, 0, \) and \( \frac{\pi}{2} \)). The location of any peak is called a peak radian. Figure 6(a) and 6(b) show three scatter plots (based on the 1st and 3rd mixtures in Fig. 1) and corresponding radian densities: when there is 1, 2 or 0 matched layer(s), in the corresponding radian density there exists the same number of clear peak(s) besides 3 axis peaks.

The clustering algorithm can be used to detect feature lines in the scatter plot with \( u \). When we have a guess on the number of matched layers, denoted by \( \hat{g} \), the line clustering algorithm with motion \( u \) and the guessed number \( \hat{g} \) is as follows.

1) Let \( S(u) = \{ ((D_k(I_1(u(x))), \nabla_k I_2(x)) \}. \) Remove the points near the origin from \( S(u) \), and then map the remaining points to the radian space: \( R(u) = \{ \text{rd} = \arctan \left( \frac{\text{pt}_2/\text{pt}_1}{\left(\text{pt}_1, \text{pt}_2 \right) \in S(u)} \right) \} \).

2) When \( n \) layers are all matched by motion \( u \), there is no cluster along any axis. To uniformly handle all situations, we deliberately add many \( -\frac{\pi}{2}, 0, \) and \( \frac{\pi}{2} \) to the radian set \( R(u) \) to avoid this case.

3) Using \( l_i^k \) and \( \mu_i \) to denote the cluster label and cluster center, implement K-means with \( \hat{g} + 3 \) centers as

\[
J(u, \hat{g}) = \min_{\{l_i^k, \mu_1, \ldots, \mu_{\hat{g}+3}\}} \sum_{\text{rd}_i \in R(u)} \left( \text{rd}_i - \mu_{l_i^k} \right)^2, \tag{27}
\]

s.t. \( l_i^k = 1, \ldots, \hat{g} + 3, \forall i \).

If our guess \( \hat{g} \) is equal to the true underlying number of matched layers, which is denoted by \( g \), for any peak radian there will be one cluster center at it. After three centers near axis radians are removed, the remaining \( g \) centers are estimated radians of the feature lines. Then mixing coefficients are gotten. However, the problem is that the matched layer number \( g \) is not known now. We will jointly consider different scatter plots with different layer motions, and present a joint clustering formulation to robustly identify feature line numbers in all these scatter plots in Section 4.5. Before that, below we discuss how to sequentially obtain all layer motions.

### 4.4 Sequential estimation of layer motions

From the properties of scatter plots, we know that most edges (i.e., significant gradients) of unmatched layers are on axes in the scatter plot, and edges of matched layers are not near any axis. When we find any correct layer motion, we can eliminate the matched edges from mixture gradients by setting the gradients not near any axis to be zero. Then, the next maximization of the motion objective function will go to the motion of another layer. Such process can be done one by one as follows.

1. Maximizing the motion objective function (21) to find a motion candidate \( c \).
2. Test line clustering with motion \( c \) and every possible cluster number \( \hat{g} \in [0, n] \), and record \( J(c, \hat{g}) \).
3. For any location \( x \) and any subscript \( k \), if \( (D_k(I_1(c(x))), \nabla_k I_2(x)) \) is close to any axis, then \( \nabla_k I_1(c(x)) \rightarrow 0 \) and \( \nabla_k I_2(x) \rightarrow 0 \).
4) Goto 1 until we have found \( n \) different candidates. Figure 6 illustrates an example. Using 1st and 3rd mixtures of Fig. 1, we maximize the motion objective function (21) as in step 1 in the above iteration and obtain the aeroplane layer’s motion. Step 2 is used to identify the matched layer number of each motion candidate, which will be discussed in the next subsection. Then we eliminate the matched gradients as in step 3, and remaining mixture gradients are shown in Fig. 6(c). The edges of the aeroplane layer have been perfectly eliminated, and the edges of other layers remain (for comparison, see Fig. 1). With such gradients, the next iteration will find another layer motion, rather than find the motion of the aeroplane again. Finally, we obtain three different motion candidates, whose corresponding scatter plots, radian densities (based on the radian set \( R \) used in the line clustering), and matched gradients are shown in Fig. 6(a), 6(b), and 6(d), respectively. Note because the vegetable layer and the Lena layer have the same motions, the second motion candidate simultaneously matches two layers. Then the third candidate can not match any layer, as main parts of significant gradients of all layers have been eliminated. In the scatter plot with such false motion, there is not any feature line.

4.5 Joint clustering

The whole layer number is known, being \( n \), and thus the number of different layer motions is no larger than \( n \). In \( n \) motion candidates, denoted by \( c_i \) (\( 1 \leq i \leq n \)), there may be some motions matching more than one layers, and there also may be some false motions. Then the next problem is how to correctly distribute \( n \) layers to these motion candidates (or distribute \( n \) feature lines to \( n \) scatter plots). Assume that the true underlying number of the layers matched by each motion candidate \( c_i \) is denoted by \( g_i \). We jointly consider \( n \) scatter plots with \( n \) motion candidates, and present a joint clustering formulation, as

\[
F_n = \min_{g_i} \sum_{i=1}^{n} J(c_i, \tilde{g}_i), \quad \text{s.t.} \quad \sum_{i=1}^{n} \tilde{g}_i = n, \quad \tilde{g}_i \in \mathbb{Z}^+ \cup \{0\},
\]

where \( J(c_i, \tilde{g}_i) \) is the optimal value (27) of line clustering and has been obtained in Step 2 of Subsection 4.4. Under the constraint of the total number being \( n \), if there is any wrong guess, then \( \exists h (1 \leq h \leq n) \), our guessed number \( \tilde{g}_h \) is smaller than the true cluster number \( g_h \), and thus in the radian set \( R(c_h) \) there must exist at least one peak radian without a very close cluster center. In such situation, \( J(c_h, \tilde{g}_h) \) will be significantly large, leading to a large objective value in (28). If our guesses are right, i.e., \( \forall i, \tilde{g}_i = g_i \), then every \( J(c_i, \tilde{g}_i) \) will be small, leading to a small objective value in (28). Consequently, the correct number of the matched layer(s) by each motion candidate can be given by the optimal solution of (28).

The problem (28) is an integer optimization problem, and can efficiently solved through dynamic programming. Consider the subproblem:

\[
F_q(r) = \min_{g_i} \sum_{i=1}^{q} J(c_i, \tilde{g}_i), \quad \text{s.t.} \quad \sum_{i=1}^{q} \tilde{g}_i = r,
\]

which denotes the minimal sum when we distribute \( r \) layers to the first \( q \) motion candidates. It can be iteratively decomposed by itself until \( q \) decreases to 1. Thereby, the iterative formulas are:

- **Initialization**: \( F_1(r) = J(c_1, r) \),
- **Iteration**: \( F_q(r) = \min_{i \in \{0, \ldots, r\}} (F_{q-1}(r-i) + J(c_i, i)) \),
- **Termination**: \( F_n = F_n(n) \),

which can be used to find the optimal value and the optimal solution of (28). The overall computational complexity of above iterations is \( O(n^3) \), which is small as \( n \) usually is a very small constant.

With the correct number of the matched layer by each motion candidate, we remove false motions and get mixing coefficients using corresponding cluster centers. Now all mixing parameters for \( n \) layers are obtained.

4.6 Extracting layer gradients

Thanks to the sparsity and independence, most of significant gradients in a mixture are contributed by only one layer. So we can approximately estimate layer gradients by assigning the 1st mixture’s gradients to them through a proper assignation method. Based on preceding analysis, for any location \( x \), in the condition of \( \nabla_k L_j(x) \) being significant, there is a high probability that the 2D point \( (\nabla_k f_1(x), D_k (I_2 f_2^{-1}(x))) \) is on a feature line with a slope equal to \( a_{2j} \). According to this property, for any given location \( x \) and subscript \( k \), we decide the assignation by minimizing radial distances between the corresponding points and feature lines, as

\[
idc(k, x) = \arg \min_j \text{rad}(k, x, j).
\]

The distance \( \text{rad}(k, x, j) \) is defined as: for \( 1 \leq j \leq n \),

\[
\text{rad}(k, x, j) = \left| \arctan \left( \frac{D_k(I_2 f_2^{-1}(x)))}{\nabla_k f_1(x)} \right) - \arctan(a_{2j}) \right|,
\]

and for \( j = n + 1 \),

\[
\text{rad}(k, x, j) = \max_{x = \frac{|ax - \frac{\pi}{2}|}{a}} \left| \arctan \left( \frac{D_k(I_2 f_2^{-1}(x)))}{\nabla_k f_1(x)} \right) - \arctan(a_{2j}) \right|.
\]

The class \( n + 1 \) is an added class to denote that the gradient does not belong to any layer. When \( \text{idc}(k, x) = n+1 \), the reason for this label is that for any \( i \), \( \nabla_k f_1(x), D_k (I_2 f_2^{-1}(x)) \) is near an axis, i.e., is not on any feature line. Thus in this case \( \nabla_k f_1(x) \) should not be assigned to any layer. When \( 1 \leq \text{idc} \leq n \), we assign \( \nabla_k f_1(x) \) to \( e^{k \text{idc}(x)} \), where \( e^{x} \) denotes the estimated gradient of the \( j \)th layer (\( 1 \leq j \leq n \)). Then we get estimated gradients of \( n \) layers, as well as remaining gradients respecting class \( n + 1 \).

Figure 6(e) shows extracted gradients of three layers from two mixtures in Fig. 1. They show approximate shapes of an aeroplane, vegetables and Lena, respectively, almost without any superimposition. Such extracted gradients will be used to match the layer order and to reconstruct original layers.
Fig. 7. Estimation of the layer number

4.7 Extension to more mixtures

Given $m$ mixtures, denoted by $I_1, \ldots, I_m$, we can apply the above sparse blind separation algorithm on two mixtures for $m-1$ times, each time on $I_1$ and another mixture $I_j$ ($2 \leq j \leq m$), and get $m-1$ groups of results. Then, we use correlations between extracted layer gradients to determine the match order of layers, and get all mixing parameters of $m$ mixtures.

5 IDENTIFYING THE LAYER NUMBER

Most previous approaches assume that the layer number is known. In practice, only mixture images are known. See the multilayer mixtures in Fig. 1. It is even not easy for human to figure out the layer number. This section discusses how to automatically identify it.

The remaining gradients (respecting the added class in Section 4.6) that are not assigned to any layer can indicate whether the layer number is correctly specified. Suppose the remaining gradients are denoted by $Grad_{left}(x, n)$ when the layer number is set as $n$, and we calculate the remaining edge quantity $Rem(n)$, as:

$$Rem(n) = \sum_x ||Grad_{left}(x, n)||_2.$$  

Then, calculate the decreasing quantity $Q(n)$:

$$Q(n) = |Rem(n-1) - Rem(n)|.$$  

$Q(n)$ denotes the quantity of newly extracted gradients. The left bar graph in Fig. 7 shows a demonstration of $Q(n)$ (based on the first two mixtures in Fig. 1, where the correct layer number is 3). $Q(n)$ is large when one true layer is extracted, and is very small after all layers have been extracted. Thus the true underlying layer number can be estimated by maximizing the following function:

$$T(n) = Q(n)/(Q(n+1)+C),$$  

where $C$ is a small quantity, and is set to be $Rem(0)/80$ in our experiments. Adding $C$ is to avoid the case that $Q(n)$ and $Q(n+1)$ are both very small but their ratio is large. The maximal $T(n)$ appear when and only when $Q(n)$ is far larger than $C$ and $Q(n+1)$ is far smaller than $Q(n)$, and thus it can be used to identify the layer number. The corresponding $T(n)$ is shown in Fig. 7. The estimated layer number is 3, the same as the true value.

To calculate the values of $Rem(n)$ and $T(n)$ for different $n$, we need to sequentially search for $n_{max}+1$ motion candidates, where $n_{max}$ denotes maximal possible layer number, and implement the joint clustering algorithm only once, with the layer number set to be $n_{max}+1$. For every $n$ ($n \leq n_{max}+1$), the optimal joint clustering objective function $F_n(n)$ as well as the optimal solution has been obtained in the iteration for $F_{n_{max}+1}((n_{max}+1)$ (See (30)-(32)). Therefore we have the mixing parameters for every $n$. Then, calculate $Grad_{left}(x, n)$ and $T(n)$ for every $n$ to identify the true layer number.

6 RECONSTRUCTION OF SOURCE LAYERS

With the mixing parameters known, the reconstruction step is the final and crucial part for the separation. Many researchers have focused on it. When motions are restricted to translations, the frequency methods are used (e.g. in [23], [24]). For more complex parameterized motions, Szeliski et al. proposed a constrained least squares formulation [20]. Nevertheless, when applying the above methods, different researchers found the “degeneracy” problem [20], [23]: although the number of different mixtures is equal or larger than the layer number, layers still can not be well reconstructed. This section will give analyses and solutions.

For simplicity, we start with a 2-layer-2-mixture example. Suppose in the 2nd mixture one layer keeps still and the other has a spatial shift $s$, and the mixing coefficients are fixed to be 1. Transfer each mixing model to frequency domain, as: at any frequency $\nu$,

$$\begin{bmatrix} f_t(I_1)(\nu) \\ f_t(I_2)(\nu) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ e^{-2\pi s^T \nu} & 1 \end{bmatrix} \begin{bmatrix} f_t(L_1)(\nu) \\ f_t(L_2)(\nu) \end{bmatrix},$$  

(39)

where $f_t(\cdot)$ denotes the Fourier transformation, and $j$ satisfies $j^2 = -1$. Note that at any frequency $\nu$ that satisfies $s^T \nu = 0$, the frequency mixing matrix is singular, and the layer frequencies are unrecoverable. The lack of layer frequencies causes damaged straight stripes along the shift direction in result layers, as shown in some results of [20], [22], [23]. For other complex motions, there also exist damaged stripes along motion directions. The substantial reason of this degeneracy problem is that the mixing model is not enough for layer reconstruction.

Besides the mixing model, the SPBS-M algorithm also offers the extracted gradients of every layer. So, we want to find the layers that agree with not only the mixing model but also the extracted gradients. Consider the reconstruction loss function:

$$J(l) = (1-\beta)\sum_{i,x} \sigma(I_i(x)) \left(I_i(x) - \sum_j a_{ij} \tilde{L}_j(f_j(x)) \right)^2 + \beta \sum_{j,x,k} \rho(|e_k^j(x)|) |\nabla_k \tilde{L}_j(x) - e_k^j(x)|,$$  

(40)

where $\tilde{L}_j$ is the reconstructed $j$th layer, and $l$ is a large vector containing all pixels of all $\tilde{L}_j$ ($1 \leq j \leq n$). The first term tends to meet the mixing model. $\sigma(y)$ is 1 when $y < 255$, otherwise is 0 (for 8-bit images). It is used to exclude saturated pixels in mixtures. The last term enforces the agreement with the extracted layer gradients $e^j(x)$, which is obtained by the method in Subsection 4.6. $\beta$ is a trade off coefficient between two terms. $\rho(y)$ is a positive and monotonously increasing function to enhance the agreement with significant gradients. In our experiments, when $y = 0$, $\rho(y)$ is 1, otherwise is 2. Most of $e_k^j(x)$ are 0 and only a few of $e_k^j(x)$ are
significantly different from 0. Thus $L_1$-norm form in the last term enforces the sparsity of layer gradients in non-edge areas but sharp edges in edge areas. Alternatively, we can replace $e^j(x)$ in (40) with zero. However, we find in such settings the results often tend to be over-smoothed. Using extracted layer gradients $e^j(x)$ avoids the smoothing force on significant gradients and can give sharper results. Note that although the $L_1$-norm loss seems to be a likelihood of Laplacian densities, theorems in Compressive Sensing [30] show that minimizing $L_1$-norm loss is suitable for all kinds of sparse signal representations, and the sparser a signal representation is, the more accurate reconstruction for the signal we can get using the $L_1$-norm loss. We have shown that image gradients are very sparse, and thus we can usually achieve good reconstruction with high image quality.

In (40), the motion is a location to location operation without any pixel value changed, the operation of mixing coefficients is only to linearly change pixel values, and the gradient operation is a combination of the vertical and horizontal difference filters. All these operations (their outputs) are linear w.r.t. $l$. So the minimization of $J(l)$ s.t. the nonnegative constraint of layer intensity can be rewritten as the following matrix form:

$$
\min_l : J(l) = (\hat{A}_r l - \delta)\top(\hat{A}_r l - \delta) + \|\hat{E}_r l - \tau\|_1, \quad (41)
$$

By introducing slack variables $w$, $\epsilon^+$ and $\epsilon^-$, the minimization problem (41) becomes:

$$
\min_{l, w, \epsilon^+, \epsilon^-} : w\top w + 1\top(\epsilon^+ + \epsilon^-), \quad (42)
$$

$$
\text{s.t.} : \hat{A}_r l - \delta = w, \hat{E}_r l + \epsilon^+ - \epsilon^- = \tau,
$$

$$
\epsilon^+ \geq 0, \epsilon^- \geq 0, l \geq 0.
$$

The above problem can be solved by quadratic programming, and the global optimal solution is obtained.

7 Simulations

In this section, we use simulated mixtures to show the effectiveness of our separation method. The settings are as follows. The planar transformation (2) is used, and the trade-off coefficient $\beta$ in reconstruction is set to be 0.01. For color images, we just use their grayscales to estimate mixing parameters in all experiments in this paper, and then $R$, $G$, $B$ channels are separately reconstructed. For good viewing, we enhance the intensities of every result image: First, we use only grayscales to calculate the normalization coefficients (a multiplier and a bias scalar). Then, we adjust the intensities of different channels by the same grayscale normalization coefficients.

A separation of 3 layers from their 3 mixtures has been illustrated in Fig. 1. To the best of our knowledge, the blind separation of mixtures containing both different mixing coefficients and different nontranslational layer motions has not yet been addressed by other methods in open literature. We now test the presented method with some changes. In the first test, we replace the joint clustering with manually setting the matched layer number by each motion to be 1. Without the joint clustering used, the vegetable and Lena layers in the 3rd mixture can not be distinguished as they have the same motion, and a false motion will be wrongly involved in. Thus we get an incorrect separation, as shown in Fig. 8(a). This test shows the necessity of the joint clustering. In the second test, we do not identify the total layer number, and deliberately specify a wrong layer number (The joint clustering is used). Figure 8(b) shows the results from the first two mixtures in Fig. 1, with the layer number wrongly set as 4. All layers have been separated, with good image quality. This test shows that our method is adaptive for the total layer number. The Bayesian approaches for ICA and BD proposed by Miskin and MacKay [31] also have such feature. Their examples are on simple layers of line drawings, whereas our method can handle more complex layers of natural scenes.

7.1 Under-determined separation

The under-determined separation is shown in Fig. 9. We mix 4 layers into only 2 mixtures, as shown in Fig. 9(a) and 9(b) (see significant rotation of the aeroplane in two
mixtures). The mixing parameters are as:

\[
A_2 = \begin{bmatrix} 0.8 & 0.86 & 1.2 & 1.08 \end{bmatrix},
\]

\[
p^{21} = [0, 3, 0.985, 0.17, 0, 0, 0, 0],
\]

\[
p^{22} = [-4, 1, 0.08, 0, 0, 0, 0, 0],
\]

\[
p^{23} = [9, -2, 1, 0, 0, 0, 0, 0],
\]

\[
p^{24} = [3, -2, 0.998, -0.05, 0, 0, 0, 0, 10^{-4}].
\]

The mixtures are quite complicated, and even human self can not easily distinguish the layer number and every object on each layer. Nevertheless, our approach correctly identifies the layer number as 4 (see Fig. 9(c)), and the well reconstructed all layers (in Fig. 9(d)), which are clear enough to show most objects, almost without any superposition.

### 7.2 Textured, Noisy, blurred and ill-conditioned mixtures

Because our approach relies on the sparsity of image gradients, one may wonder how it works on texture images, whose gradients seem not that sparse. We use a complex grass texture image and a complex leaves texture image as original layers, as shown in Fig. 10(a). To make the problem more challenging, we superimpose these layers with a singular all-ones mixing coefficient matrix, in addition to different planar motions (see the horizontal or vertical scalings of two layers in different mixtures), as shown in Fig. 10(b). Our approach gives a correct layer number (see Fig. 10(c)) and correct layers with high image quality (see Fig. 10(d)). This experiment validates that our method is robust to both complex textures and singular mixing coefficients.

Here we explore the boundary of our algorithm using noisy, blurred and ill-conditioned mixtures. To make the problem slightly easier, in this subsection we manually set the layer number. First, we mix two layers into four mixtures with different layer motions and mixing coefficients, and add the mixtures with Gaussian white noises whose variance is equal to 0.004 (the image intensities are within [0, 1]), as shown in Fig. 11(a). In the noisy case, a larger \(\beta\) is needed to enforce smoothness and mitigate noises, and we set \(\beta = 0.0025m\). The separations using the first two mixtures and all four mixtures are shown in Fig. 11(b) and Fig. 11(c), respectively. Our algorithm is robust to such noises, and the more mixtures we use, the better image quality we get. When we increase the variance of the noise to 0.02, our method only finds correct motions of one layer, but fails in finding correct motions of the other layer as the gradients of the firstly matched layer can not be accurately eliminated in the very noisy case. Second, we mix two layers into two mixtures with different motions, different mixing coefficients, and different convolutions with the Gaussian kernels. The standard deviations of the Gaussian kernels for the aeroplane layer in two mixtures are 2 and 3 (pixels), respectively, and the ones for Lena layer are 2.5 and 1.5, respectively. The mixtures are shown in Fig. 11(d). On such blurred mixtures, our method also achieves an acceptable separation, although does not perform the exact deconvolution on each layer, as shown in Fig. 11(e). When we set largely different Gaussian kernels of a same layer in different mixtures (e.g., the standard deviation difference is larger than 5), our algorithm does not give a good separation. Finally, we consider ill-conditioned mixtures. We mix two layers into two mixtures, and constrain that two layers have the same motions in each mixture. In such case, there exist two feature lines in one scatter plot. We further make the mixing coefficient matrix to be approximately singular, then the feature lines will be nearly co-linear. The mixtures with \(A_2 = [0.94, 1.06]\) and their separation (with \(\beta = 0\)) are shown in Fig. 11(f) and Fig. 11(g), respectively. In ill-conditioned cases, the reconstruction result is sensitive to noises, and even to truncation errors. We further set
8 SEPARATING REAL SUPERIMPOSED IMAGES

To further demonstrate the performance of our method (referred to as SPBS-M), we use real mixture images containing reflections and transparency, and the real crossfade images recorded from a TV programme. For comparison, we also apply other blind separation algorithms: fastICA [32] and 2D-AC-DC [22]. The settings in the experiments are as follows. (1) In SPBS-M, the layer number is automatically identified, and the trade-off coefficient $\beta = 0.02$. In other methods the layer number can not be estimated and is set manually. (2) The planar transformation (2) is used in SPBS-M. (3) A derivative filter is performed as preprocessing steps of fastICA and 2D-AC-DC, as in [22], [33], and the initial guesses in 2D-AC-DC are the same as in [22].

8.1 Transparency and reflection

Fig. 12(a) shows two real world photos containing a transmitted layer of a painting and another reflected layer of an outside scene. When one meets such mixing problem in his photo, one of the most convenient way for separation is to take another shot after a movement. Without enough preparation, normal users will almost inevitably introduce some rotations or scalings besides translations, due to hand jitter or some other factors. As shown, our photos contain some translations and some rotations. With such two mixtures, fastICA does not achieve correct separation, and gives results which are almost the same as original mixtures (see Fig. 12(c)). 2D-AC-DC extends ICA to shifting mixtures. It also fails in the separation as it can not deal with rotations (see Fig. 12(d)). SPBS-M successfully recognizes the layer number as 2 (Fig. 12(b)), and finds accurate layer motions. Then SPBS-M gives two clear layers, where there is almost no superposing effect (Fig. 12(h)).

For a more complete comparison, we also implement the mixture motion method of Jepson and Black [14] (referred to as MM) for motion estimation, and use the single-image user-assisted separation system by Levin and Weiss [3] (SIUA) and the constrained least squares reconstruction algorithm by Szeliski et al. [20] (CLS) for reconstruction. First, we perform CLS using the motions output by SPBS-M. Note that each layer has little intensity change in different photos, and in such case there exists the degeneracy problem in the mixing model. Although CLS can give good results when there are a large amount of mixtures with motions of various directions, in the two-photo case it offers two layers with obvious stripes because of the degeneracy (Fig. 12(g)). Our reconstruction method perfectly addresses the degeneracy problem, as shown in 12(h). Then, we try a combination of MM and SIUA. We implement MM with random initial values for 20 times and record the most frequently occurring result. To check local minimal problems, we also use the motions output by our method as initial values in MM. In such settings, MM still outputs the same motions as the most frequently occurring result. So this most frequent result is selected as final output of MM. Based on this final output and a default all-ones mixing coefficient matrix, our gradient extraction approach is utilized to assign labels of mixture gradients. With these labeled gradients, SIUA is applied on the 1st mixture for the separation. The results of such combination method are shown in Fig. 12(e). Unfortunately, MM does not give correct layer motions as its motion estimation from superimposed images is disturbed by both textures in two layers and the relatively large motion of the painting layer (See the explanation in Section 4.2. Besides the first order Taylor estimation of brightness constancy used in MM may also be not accurate enough due to large motions). Therefore such combination does not achieve a successful separation. Finally, we use SPBS-M for motion estimation and gradient extraction and SIUA for reconstruction. As SPBS-M gives correct motions and correct layer gradients, SIUA offers an approximate separation, as shown in Fig. 12(f). Nevertheless, since only one mixture is utilized in SIUA, the image quality of their results is not very high. By exploiting all mixtures, our reconstruction method provides layers of higher quality, as shown in Fig. 12(h).

$$A_2 = [1, 1 + 2 \times 10^{-7}]$$, and then our algorithm can not accurately estimate the mixing coefficients with 256-levels truncated intensities of mixtures. However, if we use the 8-bytes double type to store mixture intensities, then the joint clustering still correctly assigns two layers to the same motion, and the estimation of $A_2$ is still accurate, equal to $[1 + 0.045 \times 10^{-7}, 1 + 1.95 \times 10^{-7}]$. 

![Real photos and layer number](image)

![Real photos](image)

![FastICA, 2D-AC-DC, MM, SIUA](image)

![CLS, SPBS-M](image)

Fig. 12. Transparency and reflections.
Fig. 13. Handling complex transformations.

We notice that there exists the color-bias problem in SIUA (see Fig. 12(e) and 12(f)). The reason is that the constant term of each channel is not recoverable from only gradients. However, we do not find any obvious color-bias in results of our method. It is because our method exploits multiple mixtures and enforces nonnegative constraints on layers’ intensities (see the constraint \( l \geq 0 \) in (41)). Szeliski et al. have shown that in some certain multiple-mixtures cases with layer motions known only correct layers without any color bias can meet the nonnegative constraints [20]. Besides, different mixing coefficients can also help. For example, if the mixing coefficient matrix is reversible, intensities including correct constant terms in different channels can be perfectly recovered.

Some photos may involve very complex transformations of layers, such as nonrigid motions and occlusions. For such images, our algorithm still can play a role of extracting the components which can be matched by the used motion model, as well as extracting the remaining component. Figure 13(a) shows four real photos of a painting in a glass frame from different viewpoints (Note, for example, the rotation of the glass frame between the first two photos), which also contain a reflected layer with very complex, occluded and nonrigid transformations, like tourists’ motions. For such images, we set the layer number to be 1, and estimate motions and mixing coefficients of the painting layer using SPBS-M. In reconstruction, we set the remaining layer in four mixtures as four different layers, each of which only appear in one mixture. So our algorithm will output five layers. As shown in Fig. 13(b), our method well separates the photos, providing high image quality.

8.2 Crossfade images

In TV and movies, crossfade effects, which are linear mixtures of fade in and fade out scenes, are widely used for scene changes. Figure 14(a) shows two crossfade images [34], where the intensity of a water-dam fade-in scene is increasing with a zoom out effect, and the intensity of the other two-engineers fade-out scene is decreasing. With such two mixtures, fastICA and 2D-AC-DC (Fig. 14(c) and 14(d)) do not achieve complete separation because they cannot handle zoom motions. Again, SPBS-M gives a correct layer number (Fig. 14(b)) and well-separated results (Fig. 14(f)). Here we also test what happens if the mixing coefficients are not correctly estimated. We use the motions output by SPBS-M and specify an all-ones coefficient matrix for reconstruction. Such method is referred to as AO, whose results are shown in Fig. 14(e). As shown, without accurate mixing coefficients, the layers cannot be well reconstructed.

Through SPBS-M, the whole crossfade process can be separated. Figure 15(a) shows 4 frames (the 1st, 4th, 6th and 9th frames) of a 9-frame crossfade process. The recovered mixing coefficients of two layers, scalings (zoom coefficients) of two layers, the fade-out video, and the fade-in video are demonstrated in Fig. 15(b), 15(c), 15(d), and 15(e), respectively. Two recovered videos are of high image quality. The demonstration video, as well as the matlab code of SPBS-M, is available at http://sites.google.com/site/gaikungk/spbsm.

9 CONCLUSION

When one meets the mixing problem in photography, one of the most convenient way for separation is to
take another shot after a movement. However, this two-
photo separation problem has not been well addressed
by previous methods. On one hand, when mixing co-
efficients are fixed, previous approaches have the de-
generacy problem. On the other hand, when the mixing
coefficients of layers change, only recent 2D-AC-DC
and SPBSS can deal with it but they are limited to
uniform translations. Our presented SPBS-M addresses
this separation problem well: it completely solves the
degeneracy problem, and can handle not only unknown
mixing coefficients but also general parameterized mo-
tions. Moreover, in our approach the layer number is au-
tomatically identified, and more layers can be separated
from fewer mixtures. The above features make SPBS-
M much more applicable than previous approaches, as
shown in experiments.

Acknowledgments
This work is supported by NSFC (Grant Nos. 60975003,
61021063 and 60605002) and 973 Program (Grant Nos.
2009CB320602 and 2010CB327904). We would like to
thank the reviewers for their constructive comments.

References
parency from the statistics of natural scenes,” in Advances in
features,” in Proc. Conf. Computer Vision and Pattern Recognition,
from a single image using a sparsity prior,” IEEE Trans. Pattern
Analysis and Machine Intelligence, vol. 29, no. 9, pp. 1647–1654,
2007.
transparent layers from a single image,” in Proc. Conf. Computer
using independent components analysis,” in Proc. Conf. Computer
decorrelation of transparent layers: The inclination angle of an
invisible surface,” in Proc. Int’l Conf. Computer Vision, 1999,
pp. 814–819.
[7] A. Cichocki and S. Amari, Adaptive Blind Signal and Image Pro-
cessing: Learning Algorithms and Applications. New York: Wiley,
2002.
[8] B. Sarel and M. Irani, “Separating transparent layers through layer
information exchange,” in Proc. European Conf. Computer Vision,
[10] A. M. Bronstein, M. M. Bronstein, M. Zibulevsky, and Y. Y. Zeevi,
“Sparse ica for blind separation of transmitted and reflected
images,” Int’l J. Imaging Science and Technology, vol. 15, no. 1,
pp. 1061–1066.
transparent and semireflected scenes,” in Proc. Conf. Computer
Vision and Pattern Recognition, 2000, pp. 38–43.
motion representation,” in the IEEE Workshop Visual Motion. IEEE,
[14] A. Jepson and M. Black, “Mixture models for optical flow com-
pilation,” in Proc. Conf. Computer Vision and Pattern Recognition,
1993, pp. 760–761.
transparent motions,” Int’l J. Computer Vision, vol. 12, no. 1,
[17] Y. Weiss and E. Adelson, “A unified mixture framework for mo-
tion segmentation: Incorporating spatial coherence and estimating
the number of models,” in Proc. Conf. Computer Vision and Pattern
motions,” Computer Vision and Image Understanding, vol. 63, no. 1,
pp. 75–104, 1996.
layer, locally affine, optical flow and regularization with transpar-
ency,” in Proc. Conf. Computer Vision and Pattern Recognition,
multiple images containing reflections and transparency,” in Proc.
Intelligence, pp. 290–301, 2006.
[22] E. Be’ery and A. Yeredor, “Blind separation of reflections with
relative spatial shifts,” in Proc. Int’l Conf. Acoustics, Speech and
[23] ——, “Blind separation of superimposed shifted images using
parameterized joint diagonalization,” IEEE Trans. Image Processing,
multiple layers with spatial shifts,” in Proc. Conf. Computer Vision
[25] ——, “Blind separation of superimposed images with unknown
motions,” in Proc. Conf. Computer Vision and Pattern Recognition,
[26] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman,
“Labelme: a database and web-based tool for image annotation.”
[27] R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. Freeman,
“Removing camera shake from a single photograph,” in ACM
introduction to compressive sampling,” IEEE Signal Processing
algorithms and applications,” Neural networks, vol. 13, no. 4-5,
[33] A. Hyvärinen, “Independent component analysis for time-
dependent stochastic processes,” in Proc. Int’l Conf. Artificial Neu-
[34] The images are from “How Water Won the West” in
TREC Video Retrieval Test Collection, at http://openvideo.

Kun Gai received his B.E. degree from Department of Automation,
Tsinghua University, Beijing, China in 2006. He is currently a Ph.D.
student in Department of Automation, Tsinghua University. His research interests include blind separation of mixture images and kernel learning.

Zhenwei Shi was a Postdoctoral Researcher in the Department of Automation, Tsinghua University, Beijing, China, from 2005 to 2007. He is currently an associate professor in the Image Processing Cen-
ter, School of Astronautics, Beihang University. His research interests include blind signal processing, image processing, pattern recognition, machine learning and neuroinformatics.

Changshui Zhang is currently a professor in the Department of Auto-
mation, Tsinghua University. He is an Associate Editor of the journal Pattern Recognition. His interests include pattern recognition, machine learning, etc.