Blind Separation of Superimposed Images with Unknown Motions

Kun Gai[1], Zhenwei Shi[2], Changshui Zhang[1]
[1] State Key Laboratory on Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology (TNList),
Department of Automation,Tsinghua University, Beijing 100084, China.
[2] Image Processing Center, School of Astronautics,
Beihang University, Beijing 100191, China.
gaik02@mails.thu.edu.cn, shizhenwei@mail.thu.edu.cn, zcs@mail.thu.edu.cn.

Abstract

We consider the blind separation of source layers from superimposed mixtures thereof, involving unknown motions and unknown mixing coefficients of layers in each mixture. Previous blind separation approaches for such problems assume motions to be uniform translations, and hence are limited for real world applications. In this paper, we develop a sparse blind separation algorithm to estimate both parameterized motions and mixing coefficients. Then, a novel reconstruction approach is presented to recover all layers, by utilizing not only the mixing model but also the statistical properties of natural images. The whole method can handle more general motions than translations, including 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 the experiments on two simulated mixtures of four layers, real photos containing transparency and reflections, and real crossfade images from videos.

1. Introduction

When we take photos through a transparent surface, like window glass, we often obtain superimposed *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 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 and recovering the component layers arises.

When only one mixture image is available, the separation is massively ill-posed (although Levin *et al*. [12, 13] attempted it and then Levin and Weiss [11] developed a two-layer separation system with user's assistances, the system is 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 the movements of the camera, the glass surface or the target object. Such movements may be inevitable (*e.g*., due to hand jitter), and can be introduced deliberately for separation. Second, layers' overall intensity may vary, since reflection angles alter along with the movements, or due to the changes of camera's settings or lighting conditions. Such diversity will lead to some changes of layers' mixing coefficients. With several different mixtures, our goal is to estimate the motions and the mixing coefficients and then recover all layers.

A number of approaches have been proposed to separate two-layer mixtures containing transparency and reflections. On one hand, some methods perform independent component analysis (ICA) [5, 6, 15, 17, 19] or utilize images with different focus [16, 18]. Static mixing is assumed in these methods. On the other hand, some technologies are proposed to recover multiple motions from image sequences, *e.g*., [4, 9, 10]. They focus on only motion recovery, without considering layer restoration. A min/max alternative method [20] is then developed to extract layers from motions (and it is extended for stereo matching in [21]). It needs a large amount of images with motions of various directions, and will meet the "degeneracy" problem [20] when there are only two or several mixtures. Moreover, it assumes fixed mixing coefficients, which can be easily violated by the changes of reflection angles or camera settings.

Recently, blind source separation (BSS) technologies show abilities to handle both relative motions and unknown mixing coefficients. Be'ery and Yeredor propose 2D-AC-DC [1, 2] to separate superimposed shifted images of two layers, and Gai *et al*. develop another fast and reliable algorithm named SPBSS [7] to handle such mixtures of multiple

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

layers without the local optimum problem. Nevertheless, these methods assume motions to be only uniform translations, and hence are limited to a small set of applications.

We develop a method to blindly separate mixtures consisting of multiple layers with unknown motions and mixing coefficients. Our method has a number of desirable properties. (1) It can deal with unknown mixing coefficients and parameterized motions, including scalings, translations, rotations and other transformations. (2) It can handle multiple layers from only two mixtures. (3) Layers still can be well reconstructed when the layer number is wrongly set by users. (4) The layer number is automatically estimated. (5) More layers can be well recovered from fewer mixtures.

Usually, the separation method contains two main parts, one of which is to estimate the mixing model and the other is to reconstruct the layers according to the mixing model. This paper has contributions on both parts. In the first part, a blind separation algorithm, named sparse blind separation with unknown motions (SPBS-M), is presented to estimate the parameterized motions and mixing coefficients. To the best of our knowledge, the separation of mixtures with unknown mixing coefficients and nonuniform motions has not yet been addressed in open literature. Then, we develop an approach to automatically identify the layer number, while in most other methods the layer number has to be specified by users. In the second part, previous reconstruction methods often have the "degeneracy" problem [2, 20], and thus can not well recover layers in some usual cases. We present a novel reconstruction approach, which takes advantages of not only the mixing model but also statistical properties of natural images, to completely solve this problem. The combination of two parts enables completely automatic separation, even when the layer number is unknown, or in underdetermined cases where mixtures are fewer than layers.

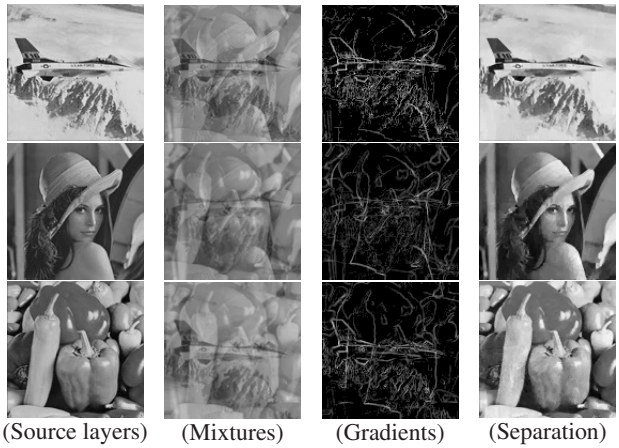
The rest of this paper is organized as follows. Section 2 presents the problem formulation. Then, Section 3 proposes SPBS-M algorithm, and the estimation of the layer number is discussed in Section 4. Subsequently in Section 5, we analyze the reason of "degeneracy", and present a novel reconstruction approach. Section 6 shows the experiments on two simulated mixtures of four layers, real photos containing transparency and reflections, and real crossfade images from a video. Finally, we close with a conclusion.

2. Problem formulation

As discussed in Section 1, in each mixture, the layers probably have unknown motions and unknown mixing coefficients. Here we assume linear mixing as in many other methods (*e.g.*, [2, 5, 7, 11, 20]), and formulate the mixing model of m mixtures with n layers as:

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

where I_i $(i = 1, \cdots, m)$ is the ith mixture, L_j $(j = 1, \cdots, n)$ is the jth 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 $f_{ij}(\cdot)$, respectively. Without loss of generality, we take the component layers in the first mixture as reference layers, and thereby $a_{1j} = 1$ and $f_{1j}(x) = x$. For simplicity, we use a matrix A to describe all mixing coefficients, and $A = [a_{ij}] = [A_1^\top, \cdots, A_n^\top]^\top$.



(Source layers) (Mixtures) (Gradients) (Separation)

Figure 1. Demonstration of the moving mixing and the sparsity.

Here only mixtures I_i are known, and the goal is to recover all model parameters and the component layers. Note that the mixing model is not space-invariant due to nonuniform motions. Such a non-space-invariant mixing problem is very challenging. We resort to the statistic properties of natural images to solve it.

Recently several studies have figured out the fact that most natural images have sparse gradients [5, 11, 12]. The sparsity denotes that only a small number of the gradients in images are significantly different from zero. The gradients significantly different from zero are called *significant* gradients, while others are called *nonsignificant* ones. The significant gradients can be seen as edges of images. It is also implied that most natural images have sparse edges.

A demonstration of the mixing model and the sparsity is shown in Fig. 1. Column 1 shows 3 layers, and column 2 shows 3 mixtures of them. The motions in the demonstration have forms of affine transformations, as:

$$f_{ij}(x) = K^{ij}\tilde{x}, \quad (2)$$

where $\tilde{x} = (x_1, x_2, 1)^\top$ is the extended location vector, and K^{ij} is a 2×3 matrix. The parameters in the 3rd mixture are as follows:

$$A_3 = (1.05, 0.575, 1.75),$$
$$K^{31} = \begin{bmatrix} 1 & -0.05 & -3 \\ 0.05 & 1 & -4 \end{bmatrix},\ K^{32} = K^{33} = \begin{bmatrix} 0.998 & 0.05 & 1 \\ -0.07 & 0.998 & 3 \end{bmatrix}. \quad (3)$$

Although the 2nd and the 3rd layers have equivalent motions, they can be discriminated owing to different mixing coefficients. Column 3 of Fig. 1 shows L_2-norms of gradients in mixtures. The gradients are sparse, as most pixels are black. Images in column 4 are the separation results from mixtures, by the proposed method in this paper.

3. Sparse blind separation with motions

This section assumes that the numbers of mixtures and layers are 2 and n, respectively, and presents an algorithm of sparse blind separation with unknown motions (SPBS-M) on 2 mixtures. Its extension to more mixtures and the automatic estimation of the layer number will be discussed in Section 3.5 and Section 4, respectively.

3.1. Hypotheses

For mathematical convenience, we use $\nabla(\cdot)$ to denote the gradient operator, and assume that the layer gradients $\nabla L_i(x)$ $(1 \leq i \leq n)$ are zero-mean processes. Then, we give the following hypotheses:

1. Sparsity: $\forall i$, ∇L_i is sparse.
2. Independence between layers: $\forall x$ and y, when $i \neq j$, $\nabla L_i(x)$ and $\nabla L_j(y)$ are independent.
3. Independence on locations: $\forall i$, when $x \neq y$, $\nabla L_i(x)$ and $\nabla L_i(y)$ are independent.

In the first hypothesis, the sparsity of gradients is found as properties of most natural images [5, 12]. Since different layers derive from different scenes, the independence between layers can be satisfied in most cases (even when the original layers are not independent, their gradients are usually independent [8]). In the perspective of filtering, the discrete gradient operator can be seen as a combination of two (vertical and horizontal) autoregressive (AR) filters, which can greatly remove the dependence of image pixels on locations [8], and thus usually the last hypothesis also tends to be met. We would like to emphasize that our algorithm is robust, as shown in experiments, even when some hypotheses are not well satisfied.

3.2. The motion object function

In order to search for correct motions, we move the first mixture with a searching motion $u(\cdot)$. Consider the following motion object function:

$$O(u) = \frac{1}{N}\left|\sum\nolimits_x \left\langle D\left(I_1\big(u(x)\big)\right), \nabla I_2(x)\right\rangle\right|, \tag{4}$$

where N denotes the pixel number, $\langle\cdot,\cdot\rangle$ represents the inner product of two vectors, and

$$D\left(I_1\big(u(x)\big)\right) = \frac{dI_1\left(u(x)\right)}{dx} = \frac{du^\top(x)}{dx}\left(\nabla I_1\big(u(x)\big)\right). \tag{5}$$

By use of the mixing model (1) and the independence between layers (assume the pixel number N is large enough), the object function in (4) can be expanded as:

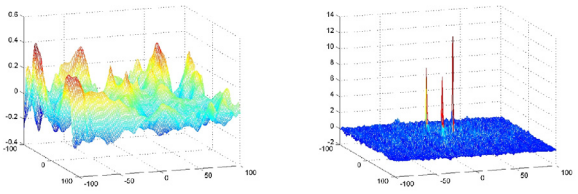
$$O(u) = \left|\sum\nolimits_j a_{2j} G_j(u)\right|, \tag{6}$$

where

$$G_j(u) = \frac{1}{N}\sum\nolimits_x \nabla^\top L_j\big(u(x)\big)\frac{du(x)}{dx^\top}\frac{df_{2j}^\top(x)}{dx}\nabla L_j\big(f_{2j}(x)\big). \tag{7}$$

On one hand, when $u(\cdot) = f_{2j}(\cdot)$,

$$G_j(u) = \frac{1}{N}\sum\nolimits_x \nabla^\top L_j\big(f_{2j}(x)\big)\frac{df_{2j}(x)}{dx^\top}\frac{df_{2j}^\top(x)}{dx}\nabla L_j\big(f_{2j}(x)\big) > 0, \tag{8}$$



NCC between images Correlation between gradients

Figure 2. The motion object function.

where the summed term is a positive determined form and the sum value will be significantly larger than zero. On the other hand, when $u(\cdot) \neq f_{2j}(\cdot)$, $\nabla L_j\big(u(x)\big)$ and $\nabla L_j\big(f_{2j}(x)\big)$ are independent, and therefore

$$G_j(u) = 0. \tag{9}$$

Being the weighted sum of $G_j(u)$, $O(u)$ is significantly larger than zero when and only when u is equivalent with a correct layer motion. Consequently, maximizing $O(u)$ can be used to search for layer motions.

We take planar and affine transformations for examples to give detailed formulations of the motion object function (4). Planar transformations can be written as:

$$\begin{aligned} u_1(x) &= p_1 + p_2x_1 + p_3x_2 + p_7x_1^2 + p_8x_1x_2, \\ u_2(x) &= p_4 + p_5x_1 + p_6x_2 + p_7x_1x_2 + p_8x_2^2, \end{aligned} \tag{10}$$

where p_1 and p_4 are translation parameters, others are respecting linear and quadratic terms. Then we get:

$$u(x) = Px^e, \frac{du^\top(x)}{dx} = X(x)P^\top, \tag{11}$$

where P is a parameter matrix, $x^e{=}(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)^T$, and

$$X(x) = \begin{bmatrix} 0 & 1 & 0 & 2x_1 & x_2 & 0 \\ 0 & 0 & 1 & 0 & x_1 & 2x_2 \end{bmatrix}. \tag{12}$$

The motion object function for planar transformations is:

$$O(P) = \frac{1}{N}\left|\sum\nolimits_x \nabla^\top I_1\big(Px^e\big)\cdot PX^\top(x)\cdot\nabla I_2\big(x\big)\right|. \tag{13}$$

Affine transformations are also widely used, which assume

$$u(x) = K\tilde{x} = K_sx + k_t, \tag{14}$$

where K_s is a $2{\times}2$ shape matrix, and k_t is a $2{\times}1$ translation vector. So, the motion object for affine transformations is:

$$O(K) = \frac{1}{N}\left|\sum\nolimits_x \nabla^\top I_1\big(K_sx + k_t\big)\cdot K_s\cdot\nabla I_2\big(x\big)\right|. \tag{15}$$

Furthermore, if we restrict motions to be translations, *i.e.*, $K_s = I$, the motion object function (15) will become the correlation between gradients. In Fig. 2, the right figure shows a motion object function in a three-layer shifted superimposed case. As discussed, the motion object function has three sharp impulses (peaks), which can be used to search for correct motions. The normalized cross correlation (NCC) between original images is also shown in the left figure. It does not have such properties, and its global maximum value does not appear at any correct motion.

The gradient-based optimization technologies are not adequate here, since the motion object function has zero gradients almost everywhere. Thereby we use discrete search

as it does not have the local optimum problem. Hierarchical methods [3] are also applied to speedup the searching. We construct an image pyramid by a desampling factor as 2. Then, discrete search is performed from smaller image level to bigger image level. At each level, we only search new occurring motions near the one respecting the optimal solution of the preceding level. Thereby, the motion number to be searched at each level is not dependent on the original pixel number N. One computation of motion object function has a complexity of $O(N)$, and the level number is $O(\log N)$. In all, the whole complexity of the hierarchical discrete search is $O(N \log N)$ (when the parameter number in the motion transformation is a small constant).

Directly maximizing the motion object function only gives one correct layer motion at the highest peak. Other parameters are still unknown. Motivated by the scatter plot method in [5, 7], we utilize sparsity to get other motions and all mixing coefficients.

3.3. Estimating all parameters via sparsity

We use the subscript k to denote the kth element of a vector, and expand derivative vectors used in (4), as:

$$\begin{cases} D_k\left(I_1(u(x))\right) = \sum\limits_{i \neq j} \frac{\partial u^\top(x)}{\partial x_k} \nabla L_i\left(u(x)\right) + \frac{\partial u^\top(x)}{\partial x_k} \nabla L_j\left(u(x)\right) \\ \nabla_k I_2(x) = \sum\limits_{i \neq j} a_{2i} \frac{\partial f_{2i}^\top(x)}{\partial x_k} \nabla L_i\left(f_{2i}(x)\right) + a_{2j} \frac{\partial f_{2j}^\top(x)}{\partial x_k} \nabla L_j\left(f_{2j}(x)\right) \end{cases}, \tag{16}$$

where two terms respecting the jth layer are listed separately for analysis. Due to the independence and sparsity, we can get properties as follows. For any location x where $\nabla L_j\left(u(x)\right)$ or $\nabla L_j\left(f_{2j}(x)\right)$ is significant (*i.e.*, significantly different from zero), there is a high probability that the gradients of other layers are nonsignificant, *i.e.*, $\nabla L_i\left(g(x)\right) = 0$ for any i ($i \neq j$) and g. It means: for most $x \in B_j \doteq \{x | \nabla L_j\left(u(x)\right)$ or $\nabla L_j\left(f_{2j}(x)\right)$ is significant$\}$,

$$\begin{cases} D_k\left(I_1(u(x))\right) = \frac{\partial u^\top(x)}{\partial x_k} \nabla L_j\left(u(x)\right), \\ \nabla_k I_2(x) = a_{2j} \frac{\partial f_{2j}^\top(x)}{\partial x_k} \nabla L_j\left(f_{2j}(x)\right). \end{cases} \tag{17}$$

On one hand, if u is the same with f_{2j}, for most $x \in B_j$,

$$\nabla_k I_2(x) = a_{2j} D_k\left(I_1\left(u(x)\right)\right). \tag{18}$$

Thus, these 2D points $\left(D_k\left(I_1\left(u(x)\right)\right), \nabla_k I_2(x)\right)$ are on a line with a slope of a_{2j}. On the other hand, when u is not the same with f_{2j}, $\nabla L_j\left(u(x)\right)$ and $\nabla L_j\left(f_{2j}(x)\right)$ are independent, and hence most significant values of $\nabla L_j\left(u(x)\right)$ and $\nabla L_j\left(f_{2j}(x)\right)$ do not appear simultaneously. Thereby, the corresponding 2D points $\left(D_k\left(I_1\left(u(x)\right)\right), \nabla_k I_2(x)\right)$ are on two axes.

For simplicity, if u is the same with a correct layer motion, we call the corresponding layer is matched by the motion u. For all k and x we plot the 2D points of $\left(D_k\left(I_1\left(u(x)\right)\right), \nabla_k I_2(x)\right)$ on a plane to get a scatter plot. *In the scatter plot, if there exist i ($0 \leq i \leq n$) layers matched by u, there will be i clusters along i 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 angular space by $\arctan\left(\nabla_k I_2(x) / D_k\left(I_1\left(u(x)\right)\right)\right)$, in the corresponding angular density plot there will be $i + 3$ peaks (Clusters along axes will cause 3 peaks at: $-\frac{\pi}{2}$, 0 and $\frac{\pi}{2}$). Two scatter plots (based on the 1st and 3rd mixtures in Fig. 1) and the corresponding angular density plots are shown in row 1 and 2 of Fig. 3: when there are one or two matched layers, in the corresponding angular density plot there exist the same number of clear peaks besides 3 axis peaks.

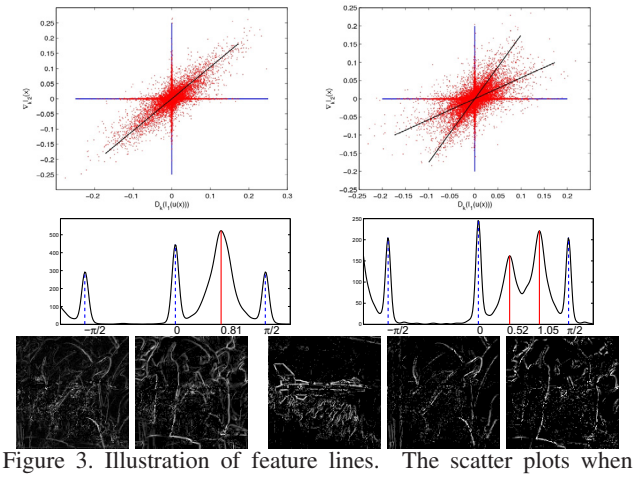


Figure 3. Illustration of feature lines. The scatter plots when one layer is matched (Left top) and when two layers are matched (Right top). The angular histograms respecting above scatter plots (The medium row). Gradients of two mixtures after the first matched layer of the aeroplane is eliminated (Left two images in the bottom row). The extracted layer gradients (Right three images in the bottom row).

From above properties, 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 axes. When we find any correct layer motion, we can eliminate gradients of matched layers from mixture gradients by setting the gradients not near axes to be zero. Then, the next maximization of the motion object function will go to the motion of another layer. Such process can be done one by one until we find n motion candidates, as follows.

1. Maximizing the motion object function (4) to find a motion candidate c.
2. Set the gradients in mixtures that are not near axes in the scatter plot of $\left(D_k\left(I_1\left(u(x)\right)\right), \nabla_k I_2(x)\right)$ to be zero.
3. Goto 1 until we have found n different candidates.

Fig. 3 shows two gradient images after one elimination. The edges of the aeroplane layer have been perfectly eliminated, and the edges of other layers remain (for comparison, the original gradients can be found in the 1st and 3rd

gradient images of Fig. 1). With such gradients, the next iteration will find the motion of other layers, rather than find the motion of the aeroplane again.

With n motion candidates, we can produce n scatter plots (when producing each scatter plot, we also can eliminate the gradients matched by other candidates to make feature lines more clear). Then, "line clustering" algorithm [7] can be used to detect scopes of feature lines matched by each motion candidates. However, the difficulty is that we do not known how many feature lines there exist in each scatter plot. Given a candidate c_i $(1 \leq i \leq n)$, we use a guess d_i as the number of feature lines, then implement line clustering on the set of points $\left(D_k\left(I_1(c_i(x))\right), \nabla_k I_2(x)\right)$, and finally obtain a sum-of-squared error $E^{c_i}(d_i)$ output by line clustering. Like in k-means, $E^{c_i}(d_i)$ has properties as follows [7]. If d_i is smaller than the true underlying number of feature lines, $E^{c_i}(d_i)$ will be significantly large, otherwise $E^{c_i}(d_i)$ will be very small. So, we can determine the number of matched layers by the following minimization problem, as:

$$\begin{aligned} \min_{d_i}: \quad & L(d_1, \cdots, d_n) = \textstyle\sum_{i=1}^{n} E^{c_i}(d_i), \\ \text{s. t.} \quad & \textstyle\sum_{i=1}^{n} d_i = n, \text{ and } \forall i,\ d_i \in \mathbb{Z}^+ \cup \{0\}. \end{aligned} \tag{19}$$

$L(d_1, \cdots, d_n)$ gets minimal value when every guess d_i is no smaller than the true underlying number. With restrictions of nonnegativity and the sum value n, the minimal value happens only when all d_i are the correct numbers.

The minimization problem (19) can be solved by dynamic programming and the global optimal solution can be found within a time of $O(n^3)$ [7]. In practice, the layer amount n is a very small constant, and $O(n^3)$ can also be seen as a small constant. With the correct numbers of feature lines, all the corresponding mixing coefficients can be given by line clustering. Now all parameters in the mixing model have been obtained.

3.4. Extracting layer gradients

Because of the sparsity and independence, most of significant gradients in the mixtures are contributed by only one layer. So we can approximately estimate gradients of layers by assigning gradients of the 1st mixture to them. Then, there is a classifying problem: which layer should we assign to? 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 point $\left(\nabla_k I_1(x), D_k\left(I_2\left(f_{2j}^{-1}(x)\right)\right)\right)$ 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 deicide the assignation by minimizing angular distances between the corresponding points and feature lines, as follows:

$$\mathrm{idc}(k, x) = \arg\min_j a_{(k,x)}(j), \tag{20}$$

where $a_{(k,x)}(j)$ is the angular distance. It is defined as: for $1 \leq j \leq n$,

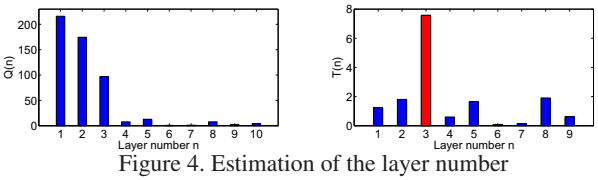


Figure 4. Estimation of the layer number

$$a_{(k,x)}(j) = \left| \arctan\left(\frac{D_k\left(I_2\left(f_{2j}^{-1}(x)\right)\right)}{\nabla_k I_1(x)} \right) - \arctan(a_{2j}) \right|, \tag{21}$$

and for $j = n + 1$,

$$a_{(k,x)}(j) = \max_i \min_{\substack{s= \\ -\frac{\pi}{2}, 0, \frac{\pi}{2}}} \left| \arctan\left(\frac{D_k\left(I_2\left(f_{2i}^{-1}(x)\right)\right)}{\nabla_k I_1(x)} \right) - s \right|. \tag{22}$$

The class $n + 1$ is an added class to denote that the gradient does not belong to any layer. When $\mathrm{idc}(k, x) = n + 1$, the reason for this label is that for any i, $\left(\nabla_k I_1(x), D_k\left(I_2\left(f_{2i}^{-1}(x)\right)\right)\right)$ is near an axis, *i.e.*, is not on any feature line. Thus in this case $\nabla_k I_1(x)$ should not be assigned to any layer. When $1 \leq \mathrm{idc} \leq n$, we assign $\nabla_k I_1(x)$ to $\nabla_k L_{\mathrm{idc}}(x)$. Then we get estimated gradients of n layers, as well as remaining gradients respecting class $n+1$.

The right three images in the bottom of Fig. 3 show the extracted gradients from two mixtures. They clearly show shapes of an aeroplane, Lena and vegetables, almost without superimposition. Such extracted gradients will be used to reconstruct original layers.

3.5. Extension to more mixtures

Given m mixtures, denoted by $I_1, \cdots, I_m$, we apply the 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 $m - 1$ results. Then extracted gradients are used to match layer orders and the whole mixing model is obtained.

4. Estimating the layer number

Almost all previous approaches assume that the layer number is known. In practice, only mixture images are known, and in those methods the layer number needs to be specified by users. See the multilayer mixtures in Fig. 1. It is not easy for human to figure out the number. This section discusses how to estimate the layer number, to make the proposed approach be a completely automatic algorithm.

The remaining gradients (respecting the added class in Section 3.4) that are not assigned to any layer can indicate whether the layer number is correctly specified. Suppose the remaining gradients are denoted by $\nabla I^r(x, n)$ when the layer number is set as n, and we calculate the remaining edge quantity $R(n)$, as:

$$R(n) = \sum_x \|\nabla I^r(x, n)\|_2, \ \ i \geq 0. \tag{23}$$

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

$$Q(n) = R(n-1) - R(n). \tag{24}$$

$Q(n)$ denotes the quantity of new extracted gradients. The left bar graph in Fig. 4 shows a demonstration of $Q(n)$ (based on the 1st and 2nd mixtures of 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:

$$T(n) = \frac{Q(n)}{\max_{k>n} Q(k)}. \tag{25}$$

The corresponding $T(n)$ is also shown in Fig. 4. The estimated layer number is 3, the same with the true value.

In other perspectives, it can also be found that SPBS-M is adaptive to the layer number. When the layer number is wrongly specified by users, layers still can be well reconstructed, which will be shown in the next section.

5. Reconstruction of source layers

With the mixing model known, the reconstruction step is the final and crucial part for the separation. Some researchers have focused on it. When motions are restricted to translations, the frequency methods are used (*e.g.* in [2, 7]). For more complex parameterized motions, Szeliski *et al*. proposed a constrained least squares formulation [20]. Nevertheless, when applying the above methods, many researchers found the "degeneracy" problem [2, 20]: 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.

5.1. Degeneracy of the mixing model

For convenience, we start at an example of two shifted layers. Suppose one layer has a spatial shift of $\mathbf{s}$, and the other layer stays static, without any mixing coefficient changed. Transfer such mixing model to frequency domain, as: at any given frequency $\mathbf{v}$,

$$\begin{bmatrix} ft(I_1)(\mathbf{v}) \\ ft(I_2)(\mathbf{v}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ e^{-2j\pi \mathbf{s}^\top \mathbf{v}} & 1 \end{bmatrix} \begin{bmatrix} ft(L_1)(\mathbf{v}) \\ ft(L_2)(\mathbf{v}) \end{bmatrix}, \tag{26}$$

where $ft(\cdot)$ denotes the Fourier transform, and j satisfies $j^2 = -1$. Note that at any frequency $\mathbf{v}$ that satisfies $\mathbf{s}^\top \mathbf{v} = 0$, the frequency mixing matrix is singular. It means the layer frequencies perpendicular to the relative shift direction are unrecoverable from the mixing model. The lack of correct layer frequencies in the perpendicular direction will cause serious straight stripes along the shift direction in the reconstructed layers, as shown in some results of [1, 2, 20].

For other complex motions, there also exist damaged stripes along motion directions (in every small patch, the motion can be approximately seen as translations, so stripes appear by the same reason). Such phenomena are called the degeneracy of the mixing model. The substantial reason is that the mixing model itself is not enough for complete reconstruction in many usual cases, like the case of two mixtures with relative motions and fixed mixing coefficients. Previous methods, which use only the mixing model, inevitably can not well reconstruct layers is such cases.

5.2. Reconstruction using extracted gradients

Besides the mixing model, the SPBS-M algorithm also offers the extracted gradients of every layer. These gradients show clear shapes of layers, and can greatly help the reconstruction. So, we want to find layers that agree with not only the mixing model but also the extracted gradients. Consider the reconstruction loss function w.r.t. layers L:

$$\begin{aligned} J(l) = (1-\beta)\sum_{i,x} s\big(I_i(x)\big)\Big(I_i(x) - \sum_j a_{ij} L_j(f_{ij}(x))\Big)^2 \\ + \beta \sum_{j,x,k} g\left(\left|e_k^j(x)\right|\right)\left|\nabla_k L_j(x) - e_k^j(x)\right|, \end{aligned} \tag{27}$$

where l is a large column vector consisting of all pixels of all layers, and $e^j(x)$ denotes the extracted gradients of the jth layers (by the method in Section 3.4). The first term tends to meet the mixing model. $s(y)$ is 1 when $y < 255$, otherwise is 0 (for 8-bit images). It is used to discard saturated pixels in mixtures. The last term enforces the agreement with the extracted gradients. $g(y)$ is a positive and monotonously increasing function to enhance the agreement with significant gradients. In our experiments, when y = 0, $g(y)$ is 1 , otherwise is 2. For most location x and subscript k, $e_k^j(x)$ is 0. So L_1-norm form in the last term also implies that the layers with sparse gradients are wanted. Finally, β is used as a trade off coefficient between two terms.

In the first term of (27), the motion is a location to location operation without any pixel value changed, and the operation of mixing coefficients is only to linearly change pixel values. Thereby the first term can be rewritten by a matrix form as:

$$(Al - b)^\top (Al - b), \tag{28}$$

where $s(\cdot)$ and $(1-\beta)$ are also integrated. In the last term, the discrete gradient operation is actually a combination of the vertical and horizontal difference filters. This term can also be transformed to a matrix form as:

$$\|El - c\|_1. \tag{29}$$

With the nonnegative constraint of layer intensity, minimization of the reconstruction loss has a formulation as:

$$\begin{aligned} \min_l &: J(l) = (Al - b)^\top (Al - b) + \|El - c\|_1, \\ \text{s. t.} &: \quad l \geq 0. \end{aligned} \tag{30}$$

By introducing slack variables w, z^+ and z^-, the minimization problem (30) becomes:

$$\begin{aligned} \min_{l,w,z^+,z^-} &: w^\top w + 1^\top (z^+ + z^-), \\ \text{s. t.} &: \quad Al - b = w, \\ & \quad El - c + z^+ - z^- = 0, \\ & \quad z^+ \geq 0,\ z^- \geq 0,\ l \geq 0. \end{aligned} \tag{31}$$

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

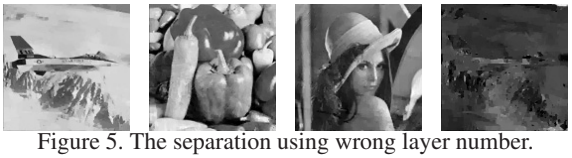


Figure 5. The separation using wrong layer number.

The presented reconstruction method not only addresses the degeneracy problem but also can recover more layers from fewer layers (will be shown in the experiments). Furthermore, the whole technologies are very adaptive to the layer number. Although the layer number can be correctly estimated by the method in Section 4, we would like to deliberately specify a wrong layer number to give an example. Fig. 5 shows the results from the mixtures in Fig. 1 which contain 3 layers, with the layer number wrongly set as 4. All layers have been separated, with good image quality. The Bayesian approach proposed by Miskin and MacKay [14] also shares the adaptive features for the layer number. Their examples are on simple layers of line drawings, whereas our method can handle more complex layers of natural scenes.

6. Experiments

In this section, we show experiments on both synthetic and real world mixtures (a separation of 3 layers from 3 mixtures has been shown in Fig. 1 and 5). First, we demonstrate an under-determined separation from two mixtures which contain four layers. Then, we perform our presented method (referred to as SPBS-M) on real world images: two photos consisting of transmitted and reflected layers, and two video images from a crossfade process.

For comparison, we also apply other blind separation algorithms: 2D-AC-DC [1] and SPBSS [7], as well as the constrained least squares reconstruction approach proposed by Szeliski *et al*. [20] (referred to as CLS-S). The settings are as follows. (1) In SPBS-M, the layer number is automatically estimated, and the trade-off coefficient $\beta = 0.7$. In other methods the layer number can not be estimated and is set manually. (2) Affine transformations (10) are used in SPBS-M. (3) A derivative filter is performed as preprocessing steps of 2D-AC-DC and SPBSS, as in [1, 7], and the initial guesses in 2D-AC-DC are the same as in [1]. (4) The real-data are color images. We only use grayscale images to estimate model parameters. Then, each color channel is separately recovered.

The under-determined separation is shown in Fig. 6. We mix four layers into two mixtures, with different mixing coefficients and affine motions as:

$$A = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.20 & 0.21 & 0.27 & 0.32 \end{bmatrix}, \quad K^{21} = \begin{bmatrix} 0.9848 & 0.1736 & -7 \\ -0.1736 & 0.9848 & 3 \end{bmatrix},$$
$$K^{22} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 2 \end{bmatrix}, K^{23} = \begin{bmatrix} 1 & 0 & 9 \\ 0 & 1 & 0 \end{bmatrix}, K^{24} = \begin{bmatrix} 0.9986 & -0.0523 & 3 \\ 0.0523 & 0.9986 & -2 \end{bmatrix}. \quad (32)$$

The mixtures (Row 2) are quite complicated, and even human self can not easily distinguish the layer number and objects on layers. However, the layer number is correctly

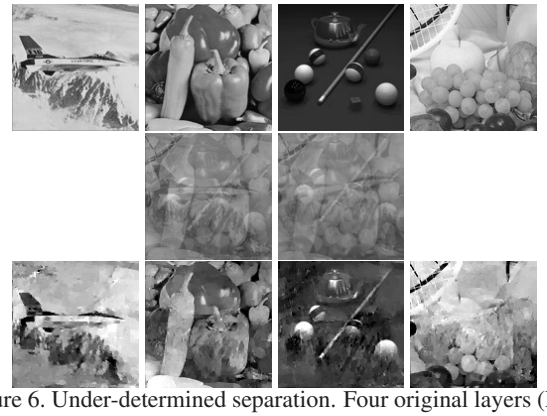
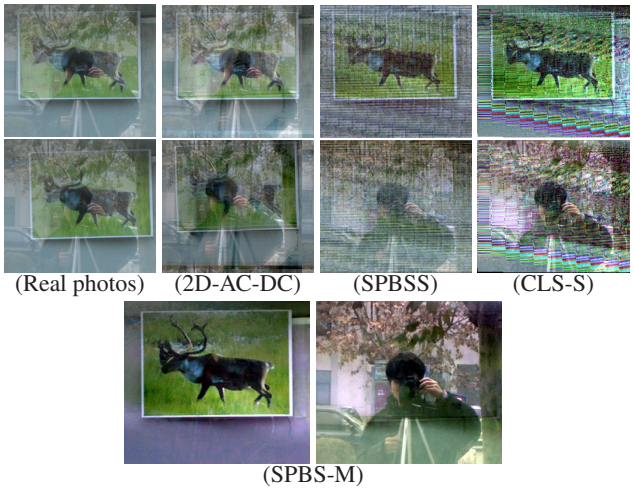


Figure 6. Under-determined separation. Four original layers (Row 1). Two mixtures (Row 2). The reconstructed results (Row 3).



(Real photos) (2D-AC-DC) (SPBSS) (CLS-S)

(SPBS-M)

Figure 7. Transparency and reflections.

estimated by our method and the reconstructed layers (Row 3) are clear enough to show most objects. To the best of our knowledge, the blind separation of mixtures containing more than two layers with nonuniform motions has not yet been addressed by other methods in open literature.

Fig. 7 shows two real world photos containing a transmitted layer of a painting and another reflected layer of a 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 slight rotations. With such two different mixtures, 2D-AC-DC and SPBSS fail in separation as they can not deal with rotations. SPBS-M successfully recognizes the layer number as 2, and finds accurate layer motions. Then SPBS-M gives two clear layers, where there is almost no superposing effect. For separately comparison of the reconstruction part, we also perform the reconstruction method by Szeliski *et al*. (CLS-S) using motions output by our method. Note that each layer has little intensity change, and in such case there exists the degeneracy prob-

(Crossfade images)

(2D-AC-DC) (SPBSS)

(SPBS-M)

Figure 8. Separating crossfade images.

lem in the mixing model. Although CLS-S can give good results when there are a large amount of mixtures with motions of various directions, in our two-photo case it offers two layers with obvious stripes because of the degeneracy. Our reconstruction method perfectly addresses the degeneracy problem, providing layers of high quality.

In TV and movies, crossfade effects, which are linear mixtures of fade in and fade out scenes, are widely used for scene changes. Row 1 of Fig. 8 shows two crossfade images[1], where a fade in scene of a water dam has increasing intensity, also with a zoom out effect, and another fade out scene of two engineers has decreasing intensity. With such two mixtures, 2D-AC-DC and SPBSS (Row 2) do not achieve complete separation because they can not handle scalings (CLS-S is not used here as it does not consider changing mixing coefficients). Again, SPBS-M (Row 3) gives a correct layer number and well-separated results. The video is recorded from an old TV programme in 1986, and the images are very noisy (See the big noisy line in the red circle on the second image). Our method is robust in such noisy case, and layers are well reconstructed.

7. Conclusion

When one meets the mixing problem in photographing, 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 other methods. On one hand, when mixing coefficients are fixed, previous approaches have the degeneracy 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 motions. Moreover, in our approach the layer number is automatically 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 the experiments.

[1]The images are from "How Water Won the West" in TREC Video Retrieval Test Collection, at http://open-video.org/details.php?videoid=489

Acknowledgment

The work was supported by the National Natural Science Foundation of China (60835002 and 60605002).

References

[1] E. Be'ery and A. Yeredor. Blind separation of reflections with relative spatial shifts. In *ICASSP*, 2006.

[2] E. Be'ery and A. Yeredor. Blind separation of superimposed shifted images using parameterized joint diagonalization. *TIP*, 17(3):340–353, March 2008.

[3] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, 1992.

[4] M. J. Black and P. Anandan. The robust estimation of multiple motions. *CVIU*, 63(1):75–104, 1996.

[5] A. M. Bronstein, M. M. Bronstein, M. Zibulevsky, and Y. Y. Zeevi. Sparse ica for blind separation of transmitted and reflected images. *IJIST*, 15(1):84–91, 2005.

[6] H. Farid and E. H. Adelson. Separating reflections and lighting using independent components analysis. In *CVPR*, 1999.

[7] K. Gai, Z. Shi, and C. Zhang. Blindly separating mixtures of multiple layers with spatial shifts. In *CVPR*, 2008.

[8] A. Hyvärinen. Independent component analysis for time-dependent stochastic processes. In *ICANN*, 1998.

[9] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *IJCV*, 12(1):5–16, 1994.

[10] S. X. Ju, M. J. Black, and A. D. Jepson. skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *CVPR*, 1996.

[11] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *TPAMI*, 29(9):1647–1654, 2007.

[12] A. Levin, A. Zomet, and Y. Weiss. Learning to perceive transparency from the statistics of natural scenes. In *NIPS*, 2002.

[13] A. Levin, A. Zomet, and Y. Weiss. Separating reflections from a single image using local features. In *CVPR*, 2004.

[14] J. W. Miskin and D. J. C. MacKay. Ensemble learning for blind image separation and deconvolution. In *ICA*, 2000.

[15] B. Sarel and M. Irani. Separating transparent layers through layer information exchange. In *ECCV*, 2004.

[16] Y. Y. Schechner, N. Kiryati, and R. Basri. Separation of transparent layers using focus. In *ICCV*, 1998.

[17] Y. Y. Schechner, J. Shamir, and N. Kiryati. Polarization-based decorrelation of transparent layers: The inclination angle of an invisible surface. In *ICCV*, 1999.

[18] Y. Y. Schechner, J. Shamir, and N. Kiryati. Blind recovery of transparent and semireflected scenes. In *CVPR*, 2000.

[19] Y. Y. Schechner, J. Shamir, and N. Kiryati. Polarization and statistical analysis of scenes containing a semireflector. *JOSA A*, 17:276–284, 2000.

[20] R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *CVPR*, 2000.

[21] Y. Tsin, S. B. Kang, and R. Szeliski. Stereo matching with reflections and translucency. In *CVPR*, 2003.