A post nonlinear geometric algorithm for independent component analysis

Nguyen, Thang Viet; Patra, Jagdish Chandra; Das, Amitabha

2005

Nguyen, T. V., Patra, J. C., & Das, A. (2005). A post nonlinear geometric algorithm for independent component analysis. *Digital Signal Processing*, 15, 276-294.

<https://hdl.handle.net/10356/94243>

<https://doi.org/10.1016/j.dsp.2004.12.006>

© 2005 Elsevier. This is the author created version of a work that has been peer reviewed and accepted for publication by *Digital Signal Processing*, Elsevier. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [DOI: <http://dx.doi.org/10.1016/j.dsp.2004.12.006>].

Downloaded on 26 Apr 2023 17:55:16 SGT

A post nonlinear geometric algorithm for independent component analysis

Thang Viet Nguyen, Jagdish Chandra Patra *, Amitabha Das

School of Computer Engineering, Nanyang Technological University, Singapore

Abstract

Simple linear independent component analysis (ICA) algorithms work efficiently only in linear mixing environments. Whereas, a nonlinear ICA model, which is more complicated, would be more practical for general applications as it can work with both linear and nonlinear mixtures. In this paper, we introduce a novel method for nonlinear ICA problem. The proposed method follows the post nonlinear approach to model the mixtures, and exploits the difference between a linear mixture and a nonlinear one from their nature of distributions in a multidimensional space to develop a separation scheme. The nonlinear mixture is represented by a nonlinear surface while the linear mixture is represented by a plane. A geometric learning algorithm named as post nonlinear geometric ICA (pnGICA) is developed by geometrically transforming the nonlinear surface to a plane, i.e., to a linear mixture. Computer simulations of the algorithm provide promising performance on different data sets.

Keywords: ICA; Post nonlinear mixture; pnGICA; Linearization

1. Introduction

Independent component analysis (ICA), which consists of recovering a set of unknown sources from their mixtures, is an important field in signal processing [1,2]. Assuming that

* Corresponding author. Fax: (65)-6792-6559.

E-mail addresses: thangnguyen@pmail.ntu.edu.sg (T.V. Nguyen), aspatra@ntu.edu.sg (J.C. Patra), asadas@ntu.edu.sg (A. Das).

the sources are independent of one another, ICA algorithm tries to find a transform of the mixtures such that the recovered signals are as independent as possible. This approach works perfectly with the classical linear ICA models, in which the mixtures are simulated by a linear mixing matrix. Linear models have been intensively studied in the last two decades and found useful in some applications [3–6].

In many practical applications, however, the sources usually undergo a nonlinear mixing process. Therefore, in such environment, the linear ICA algorithms do not provide a reasonable solution. One of early approaches to nonlinear ICA problem was proposed in Refs. [7,8], using self organizing maps model to separate the mixtures, but suffered from high computational complexity. Moreover, without additional constraint to the nonlinear mixing model, there exists an infinite number of solutions for nonlinear ICA [9,10]. The problem can be overcome by exploiting different constraints beyond the independence, for example, extracting sources with bound distribution [11], exploiting the temporal structures of the time correlated sources [4,12], or working on nonlinear functions whose inverses are assumed to be well approximated by a priori artificial neural network (ANN) [13,14].

In this work, we follow a well-known post nonlinear (PNL) mixtures model [15], an approach that constrains the nonlinear process to be univariate functions. In PNL mixing scheme, the unknown sources are first linearly mixed, then each of the mixed signals is put through a univariate invertible nonlinear function. The final outputs therefore are called post nonlinear mixtures. Besides its theoretical interest, this model is suitable for many real-world applications such as sensor array processing [16], digital satellite and microwave communications [17], and in biological systems [18]. This approach has been adopted for many nonlinear models like those in Refs. [15,19–21].

The PNL separation is a two-stage process, a nonlinear transformation followed by a linear demixing process. The linear demixing is simply a classical linear ICA model, and most of the work is focused on the nonlinear transformation, or linearization. In Ref. [15], Taleb et al. propose a method that applies a multi-layer perceptrons (MLP) network to approximate the nonlinear function. A similar MLP-based approach with different independent criterion is proposed in Ref. [21]. Ziehe’s approach [19,20] presumes the linear mixture has normal distribution, and applies this assumption to estimate the outputs of the linearization. However, most of the reported algorithms just consider a nonlinear mixture as the result of a univariate function of the linearly mixed signal, not as the result of a multivariate function of the original sources. Hence, some important linear properties have not been exploited.

Our proposed algorithm approaches the problem from a geometric viewpoint, exploiting the difference between the distribution of a linear mixture and that of a nonlinear one. Preliminary result of our method has been reported in Ref. [22]. In a multidimensional space, a linear mixture is represented by a plane while a nonlinear mixture is generally represented by a curved surface. The nonlinearity of a mixture will vanish when the mixture’s representation, i.e., a surface, is transformed to a plane. Based on this idea, our algorithm first represents the distribution of an input in a multidimensional space, then geometrically transforms it to a plane, and finds the linear mixture corresponding to that plane. Finally, the extracted linear mixtures undergo a linear demixing process to estimate the original source signals. Comparing with other methods, our method has the following advantages:

- The proposed method does not require any additional assumption about the inputs, their distribution or the nonlinear functions.
- The linearization is performed directly on the data and does not need complex computation.
- The linear demixing process can be chosen from any suitable linear ICA method.
- The extracted signals are easy to visualize and to verify their linearity.

2. Post nonlinear mixtures

The post nonlinear mixtures approach [15], like any ICA model, includes two components, the mixing system and separating system (Fig. 1). In mixing system models, m unknown source signals, represented by a vector $\mathbf{s} = [s_1(t), s_2(t), \dots, s_m(t)]^T$, are linearly mixed by a mixing matrix $\mathbf{A}_{n \times m}$ to produce intermediate linear mixtures given by $\mathbf{v} = [v_1(t), v_2(t), \dots, v_n(t)]^T$. Symbol t denotes discrete time index. The linear mixing process can be modeled as

$$\mathbf{v} = \mathbf{A}\mathbf{s}. \quad (1)$$

Each intermediate mixture $v_i(t)$ is then nonlinear transformed by a nonlinear function f_i to produce the nonlinear observation $\mathbf{x} = [x_1(t), x_2(t), \dots, x_n(t)]^T$. The transform process is given by

$$x_i(t) = f_i(v_i(t)), \quad i = 1, \dots, n, \quad (2)$$

where f_i are unknown invertible nonlinear functions.

The separating system, which can be viewed as an inverse of the mixing system, includes two stages: a linearization stage followed by a linear demixing stage. The former converts the nonlinear observed signals $\mathbf{x}$ to linear mixtures, $\mathbf{z} = [z_1(t), z_2(t), \dots, z_n(t)]^T$. It can be done by estimating a function g which is an inverse of the unknown nonlinear function f (i.e., $g = f^{-1}$). The demixing stage takes these linear mixtures $\mathbf{z}$ as inputs and applies a linear ICA algorithm to extract the original signals. The separating system is modeled as

$$z_i(t) = g_i(x_i(t)), \quad i = 1, \dots, n, \quad (3)$$

$$\mathbf{y} = \mathbf{W}\mathbf{z}, \quad (4)$$

Fig. 1. Block diagram of the PNL mixture and separation system.

where $\mathbf{W}_{m \times n}$ is called the demixing matrix and $\mathbf{y} = [y_1(t), y_2(t), \dots, y_m(t)]^T$ is the vector of final outputs. As it can be seen, the linearization stage is the critical part in the PNL model. More details on linear ICA algorithms (used in demixing stage) can be found in Ref. [2].

3. The proposed geometric approach

In most reported PNL algorithms, function f_i is considered as a univariate function of the linear mixture v_i , $i = 1, \dots, n$. Therefore, the differences between a linear mixture and a nonlinear mixture are not fully exploited. In this section, we introduce an approach that further exploits the geometric difference between these two kinds of mixtures. We consider f_i as a multivariate function of the source signals, s_j , $j = 1, \dots, m$. For a simple explanation, we will detail the algorithm in the case of two sources only. The solution for more than two sources can be extended from the algorithm for two sources.

In a three-dimensional space, a surface S can be represented by

$$\gamma = f(\alpha, \beta), \quad (5)$$

where α, β, γ are three axes and f denotes an arbitrary function. A plane V , which is a special case of the surface, is represented by

$$\gamma = a\alpha + b\beta + c, \quad (6)$$

where a, b, c are arbitrary constants. If by some geometric operators, the surface S is transformed into a plane V , then the nonlinear mixture γ (5) will be converted into a linear one (6).

In the case of only two sources, Eqs. (1) and (2) can be written as

$$v_1(t) = a_{11}s_1(t) + a_{12}s_2(t), \quad (7)$$

$$v_2(t) = a_{21}s_1(t) + a_{22}s_2(t), \quad (8)$$

$$x_1(t) = f_1(a_{11}s_1(t) + a_{12}s_2(t)), \quad (9)$$

$$x_2(t) = f_2(a_{21}s_1(t) + a_{22}s_2(t)). \quad (10)$$

Let s_1, s_2, x_i , $i = 1, 2$, represent the axes α, β , and γ , respectively. From (5), the observed signal x_i (9) and (10) are represented by a surface S_i . The conversion of x_i to a linear mixture v_i is equivalent to a transformation of S_i to a plane V_i . It could be quite easy if all the values s_1 and s_2 and the corresponding values x_i are available. In our PNL problem, however, neither s_1 nor s_2 are known. Hence, to accomplish the conversion, we utilize the following properties of the midpoint and the time index.

Proposition 1. *Let (p_1, p_2) be a pair of two arbitrary points on a surface S in 3-D space. Let $\overline{p_1 p_2}$ be a line joining p_1 and p_2 and p_m be the midpoint of $\overline{p_1 p_2}$. Then the surface S is a plane if and only if for every pairs (p_1, p_2) , the point p_m also lies on S .*

An illustration of the Proposition 1 is shown in Fig. 2. Since p_{1v} and p_{2v} are on plane V , the midpoint p_{mv} of $\overline{p_{1v} p_{2v}}$ lies on V . Whereas, as S is a surface, the midpoint p_{ms} of $\overline{p_{1s} p_{2s}}$ falls out of S with $p_{1s}, p_{2s} \in S$.

Fig. 2. The plots showing difference between (a) plane V and (b) surface S . The midpoint p_{mv} of $\overline{p_{1v}p_{2v}}$ is on V while the midpoint p_{ms} of $\overline{p_{1s}p_{2s}}$ falls out of S .

Since the values of s_1 and s_2 (i.e., the value on α and β axes) are unknown, we cannot identify the absolute position of a point. Instead, we use the time index to identify the relative position between two points. This property is provided by the following proposition:

Proposition 2. For any two points p_1, p_2 that represent for $x_i(t_1)$ and $x_j(t_2)$ (i.e., the γ -axis value), if $t_1 = t_2$ then p_1 and p_2 have the same α, β -coordinates. p_2 is called the companion point of p_1 , and the pair $(p_1, p_2)_t$ is called a companion pair.

The idea of our geometric-based approach is to iteratively move each point on the surface S_i to the right position on a plane V_i . First, we select two arbitrary points $p_{1s}, p_{2s} \in S_i$, find out the midpoint p_{ms} of the line $\overline{p_{1s}p_{2s}}$, identify the companion point p_c of p_{ms} , with p_c lies on the surface S , then move the point p_c to the position of p_{ms} . By doing that, p_{1s}, p_{2s} , and p_c are placed on a same plane. Figure 3 illustrates the position of p_c before and after the transformation.

The other problem, however, is to find out the position of the point p_c on S . The following proposition provides the relationship among the companion pairs which is used in our geometric-based method to identify the point position:

Proposition 3. Given two planes V_1 and V_2 , let p_1, p_2 be two arbitrary points on V_1 and q_1, q_2 be two points on V_2 such that $(p_1, q_1)_{t1}$ and $(p_2, q_2)_{t2}$ are companion pairs, respectively. If p_m is the midpoint of $\overline{p_1p_2}$ then its companion point q_m will be the midpoint of $\overline{q_1q_2}$.

A perspective view of our method is shown in Fig. 4. Assuming that there is a plane V , which created from (7) or (8), whose all point values are known (the values on γ -axis). First, select two arbitrary points p_1 and p_2 on the surface S , and let p_m be the midpoint of $\overline{p_1p_2}$. Second, locate the companion points q_1 and q_2 on the plane V of p_1 and p_2 . Third, compute the midpoint q_m of $\overline{q_1q_2}$. Clearly, q_m is on V since V is a plane (Proposition 1)

Fig. 3. An iteration of midpoint transformation progress, a point on the surface is moved to a plane. Point p_c (a) before and (b) after transformation.

Fig. 4. Linearizing the surface S by a reference plane V . Point p_c (a) before and (b) after transformation.

and $(p_m, q_m)_{tm}$ is a companion pair (Proposition 3). Finally, find the companion point q_c on S of q_m and change p_c to p_m . The plane V is called ‘reference plane.’ In summary, the scheme to transform a surface to a plane using a reference plane is

- (1) Select two random points p_1 and p_2 on the surface S .
- (2) Calculate the midpoint p_m of the line $\overline{p_1 p_2}$.
- (3) Find the points q_1, q_2 on reference plane V such that $(p_1, q_1)_{t1}$ and $(p_2, q_2)_{t2}$ are two companion pairs.
- (4) Calculate the midpoint q_m of the line $\overline{q_1 q_2}$.
- (5) Find the point p_c on S such that $(q_m, p_c)_{tm}$ is a companion pair.
- (6) Change p_c to p_m .
- (7) Repeat (1) to (6).

4. The pnGICA algorithm

The above geometric-based approach works when the reference plane is available. In a real PNL model, however, we have to overcome the following problems: (i) there is no reference plane to use and (ii) we cannot identify the absolute position of the midpoint q_m since the values on α, β -axes (the sources s_1, s_2) are unknown.

The first problem can be treated by using a ‘fake plane,’ i.e., presuming another surface as the reference plane. For the second problem, we apply a heuristic criterion for selecting the midpoint: assuming that $v(t)$ is the value on γ -axis of point q_t , the following equation holds true:

$$v(t_m) = \frac{v(t_1) + v(t_2)}{2}. \quad (11)$$

Thus, we search for a point satisfies (11), presuming it to be the midpoint.

The utilization of fake plane and heuristic criterion results in some errors during the transformation. For example, when there are many points satisfying the criterion (11), the algorithm may select a wrong midpoint q_m and lead to a false position of p_c . Hence, we apply a learning rate μ during the updating process. Moreover, after the transformation, we employ a smoothing process on the outputs, z (the linearized signals), to reduce the errors. The smoothing function used in this work is formulated as

$$z(t) = \sum_{j=-L}^L z(t+j)/(2L+1), \quad (12)$$

where L is called the window length. This function simply replace the signal value at time t by the average of the all the values inside a window centered at t . For more discussion on different smoothing techniques (see Refs. [23–25]).

The pnGICA algorithm for PNL model of n observations, $x_1, \dots, x_n$, is similar to that of two sources. Each input signal x_i is presented by a surface S_i that needs to be transformed. Instead of fixing a surface as the fake plane to update the others, pnGICA alternatively selects one among these surface S_i , and update the rest $n-1$ surfaces by the selected one. Figure 5 illustrates an iteration of our pnGICA algorithm. Figure 5a displays a selected point and Fig. 5b shows the position of this point after updating.

In summary, pnGICA algorithm attempts to convert a surface into a plane. This is equivalent to converting a nonlinear mixture $x_i(t)$ into a linear mixture $z_i(t)$. The linear mixtures $\mathbf{z} = [z_1(t), z_2(t), \dots, z_n(t)]^T$ are then processed by a linear ICA algorithm. Finally, the linear ICA algorithm, acting as the second stage, produce the separated signals $\mathbf{y} = [y_1(t), y_2(t), \dots, y_m(t)]^T$. The following paragraph provide a framework of pnGICA method.

The pnGICA algorithm

Input

N_b /* number of iterations of the outer loop */
 N_s /* number of iterations of the inner loop */
 μ /* the updating rate */

Fig. 5. An example of the algorithm. $x_1(t)$ is linearized by using $x_2(t)$ as the reference plane. (a) Position of p_c before the updating process. (b) p_c after updating with learning rate $\mu = 0.5$.

```

    L    /* the smoothing window length */
    x    /* the observed signals  $\mathbf{x} = [x_1(t), x_2(t), \dots, x_n(t)]^T$  */
Output
    z    /* the linearized signals  $\mathbf{z} = [z_1(t), z_2(t), \dots, z_n(t)]^T$  */
function z = pnGICA(y,  $N_b$ ,  $N_s$ ,  $\mu$ ,  $L$ )
{
    /* initiate the variables */
     $z_i(t) = x_i(t)$ ;    /*  $i = 1, 2, \dots, n$  */

    /* Iteratively choose 1 signal as the fake plane and update the rest */
    For  $c_1 = 0$  to  $N_b$  do
    {
        Choose the index  $k$  randomly;

        /* Iteratively select the points and update their positions */
        For  $c_2 = 0$  to  $N_s$  do
        {
            Choose  $t_1$  and  $t_2$  randomly;
            Find  $t_m$  so that  $z_k(t_m)$  satisfies (11);
            For  $i = 1$  to  $n$  do
            {
                /* update the values for  $z_i(t_m)$  */
                 $y = (z_i(t_1) + z_i(t_2))/2$ ;
                 $z_i(t_m)^{\text{new}} = (1 - \mu)z_i(t_m)^{\text{old}} + \mu y$ ;
            }
        } /* end of inner loop */

        /* Smoothen the modified signals */
        For  $i = 1$  to  $n$  do
        {

```

```

        Smoothen  $z_i(t)$  using (12);
    }
} /* end of outer loop */
} /* end of the function */

```

5. Simulation results

To evaluate the performance of the proposed pnGICA, several computer simulations have been carried out on different data sets. The first test was done on a simple case of two sinusoidal sources in order to make an easy visualization and a detail analysis. The other two tests, which were accomplished with speeches and images, are to assess the ability of pnGICA in multiple sources PNL problem.

The original sources were linearly mixed by a random mixing matrix $\mathbf{A}$ (1) and then transformed one-by-one by a set of nonlinear functions f_i (2) to produce the post nonlinear mixtures. These mixtures were the inputs for our pnGICA algorithm. The outputs of pnGICA, the linearized signals, were put through a linear ICA algorithm to separate the signals.

In order to give a comparative view of the pnGICA performance, we repeated the same test on a linear ICA algorithm, i.e., run the linear ICA method on the nonlinear mixtures directly, and compare the final outputs. In addition, we also compare our results with those of another well-known PNL algorithm, the Gauss-TD [19]. The performance were measured by the correlation coefficient index r . The correlation coefficient between an original source $s(t)$ and an output signal $y(t)$ is computed by

$$r = \frac{\sum_{t=1}^N (s(t) - \bar{s})(y(t) - \bar{y})}{\sqrt{\sum_{t=1}^N (s(t) - \bar{s})^2 \sum_{t=1}^N (y(t) - \bar{y})^2}}, \quad (13)$$

where $\bar{s} = \frac{1}{N} \sum_{t=1}^N s(t)$, $\bar{y} = \frac{1}{N} \sum_{t=1}^N y(t)$, and N is the number of the samples. The SOBI algorithm [5] has been chosen as the linear ICA algorithm to perform the comparing test as well as to separate the signals from the linearized outputs of pnGICA (i.e., to perform the second stage in PNL separating model).

5.1. Experiment 1. Mixture of two sinusoidal signals

The observation signals $x_1(t)$ and $x_2(t)$ were generated from two sinusoidal signals by the PNL mixing scheme shown in Fig. 1. Entries of mixing matrix $\mathbf{A}$ were random numbers in range of $[-1, 1]$. The 3D plots of the linear mixture $v_i(t)$ and nonlinear mixture $x_i(t)$ are shown in Fig. 6. The purpose of pnGICA is to obtain the linearized signals $z_i(t)$ from the inputs x_i so that the 3D presentations of $z_i(t)$ look like the 3D presentations of $v_i(t)$ as much as possible (i.e., a plane in 3D space).

$$s_1(t) = \sin(0.33t), \quad s_2(t) = \sin(2\pi(0.3t)) + 6 \cos(2\pi(0.06t)), \quad (14)$$

$$\mathbf{A} = \begin{pmatrix} 0.3553 & -0.0172 \\ -0.6693 & 0.2001 \end{pmatrix}, \quad (15)$$

Fig. 6. Plots of the linear mixtures $v_i(t)$ and the nonlinear mixtures $x_i(t)$, $i = 1, 2$. The signals were rescaled between $[-1, 1]$. (a) $v_1(t)$ and $v_2(t)$. (b) $x_1(t)$ and $x_2(t)$.

Fig. 7. Plots of the unknown linear mixtures $v_i(t)$ and the linearized signals $z_i(t)$. The signals were rescaled between $[-1, 1]$. (a) $z_1(t)$ and $z_2(t)$. (b) $v_1(t)$ and $v_2(t)$.

$$f_1(v) = v^3, \quad f_2(v) = \tanh(4v). \quad (16)$$

The pnGICA algorithm was set with 1000 iterations for the outer loop ($N_b = 1000$) and 50 for the inner loop ($N_b = 50$). Learning rate μ and window length L of smoothing function were set to 0.2 and 50, respectively. The linearized signals $z_i(t)$ are shown in Fig. 7a, next to the unknown linear mixture $v_i(t)$ (Fig. 7b). As we can see, the 3D plots of $z_i(t)$ are almost identical to those of $v_i(t)$, showing clearly that the nonlinearity of x_i has been removed out of z_i . In other word, the pnGICA algorithm is capable of estimating the underlying linear mixtures from the observed nonlinear mixtures.

Finally, we evaluate the pnGICA performance by comparing the final outputs $y_i(t)$ with the original sources s_i , $i = 1, 2$, in term of the correlation coefficient r . The values of r are provided in Table 1 in which a bold number represents for the correlation coefficient

Table 1

Experiment 1. Mixture of two sinusoidal signals—correlation coefficient r between the original sources and the extracted signals. (Bold number represents correlation coefficient between a source and its corresponding estimate)

	s_1	s_2
SOBI		
y_1	-0.9831	0.1481
y_2	0.0316	0.1466
Gauss-TD		
y_1	-0.9404	0.1605
y_2	-0.0261	0.8308
pnGICA		
y_1	-0.9994	0.0114
y_2	-0.0007	-0.9209

between a source signal and its corresponding estimate. Comparing to the result obtained directly from a linear ICA algorithm (the SOBI algorithm as in this experiment), pnGICA result is quite good with higher correlation coefficient values. A slightly better performance is also observed when we compare with Gauss-TD algorithm [19].

The outputs signals y_i are plotted in Fig. 8, against the original sources, the nonlinear mixtures and the comparing SOBI outputs and Gauss-TD outputs. As it is shown in Fig. 8, pnGICA has successfully extracted the two original sources while the linear ICA (SOBI) could manage to recover only one.

Moreover, to verify the performance of pnGICA when the nonlinear functions are the same, i.e., $f_1 = f_2$, we repeated the Experiment 1, but this time, the two functions were chosen to be the same: $f_1(v) = f_2(v) = \tanh(7v)$. The results are displayed in Fig. 9 and the correlation coefficients r are provided in Table 2. Clearly, these results show that pnGICA performance was not affected in the case the same nonlinear functions were selected.

5.2. Experiment 2. Mixture of four speech signals

In this simulation, 5000 samples of four speeches (taken from [26]) were chosen as the original sources. They were linearly mixed by a random matrix $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} -0.7056 & -0.2723 & -0.8562 & -0.4275 \\ 0.3966 & 0.1866 & -0.2596 & 0.0014 \\ -0.7758 & 0.7012 & 0.1318 & -0.4642 \\ 0.8582 & -0.2290 & 0.3806 & -0.8994 \end{pmatrix} \quad (17)$$

and then nonlinearly transformed by the following functions $f_i, i = 1, 2, 3, 4$:

$$\begin{aligned} f_1(v) &= \tanh(2v), & f_2(v) &= v + 0.1v^3, \\ f_3(v) &= v^3, & f_4(v) &= v + \tanh(3v). \end{aligned} \quad (18)$$

The nonlinear mixtures, i.e., the observed signals are shown in Fig. 10, together with the original speeches. The separating system, the pnGICA, was configured with $N_b = 2000$,

Fig. 8. Plots of (a) the two unknown original sources $s_1(t)$ and $s_2(t)$, (b) their nonlinear mixtures $x_1(t)$ and $x_2(t)$, (c) the SOBI result, (d) the Gauss-TD result, and (e) the pnGICA result.

Table 2

Experiment 1a. Mixture of two sinusoidal signals with the same nonlinear functions—correlation coefficient r between the original sources and the extracted signals. (Bold number represents correlation coefficient between a source and its corresponding estimate)

	s_1	s_2
pnGICA		
y_1	-0.9983	-0.0159
y_2	-0.0011	-0.9052

$N_s = 50$, $\mu = 0.2$, $L = 5$, and SOBI as the linear demixing algorithm. The pnGICA results, are again compared with those of SOBI [5] and Gauss-TD [19], and plotted in Fig. 11.

A very good performance of pnGICA is evident in this figure: the linear ICA was not suitable for separating nonlinear mixtures. It could estimate only one speech with ad-

Fig. 9. The result of pnGICA in case when nonlinear functions are the same. (a) The two linearized signals $z_1(t)$ and $z_2(t)$ and (b) the final outputs $y_1(t)$ and $y_2(t)$.

Fig. 10. Plots of (a) the original sources and (b) the PNL mixtures.

equate quality but failed to extract the rest three speeches. Whereas, the pnGICA was capable of estimating all the four speech signals effectively. The correlation coefficients between outputs $y_i(t)$ and the original signals $s_i(t)$ for the linear ICA algorithm (SOBI), comparing PNL method (Gauss-TD) and pnGICA are provided in Table 3. In the case of linear ICA, the correlation coefficients of its outputs and the original signals drop drastically down to $r = 0.4551$. Whereas, our proposed algorithm continues to keep a good result with high quality estimated signals, even the poorest output of pnGICA still has the coefficient value over 0.935, i.e., it is clearly separated from mixtures. Moreover, comparing with the Gauss-TD algorithm, which uses additional assumption of the Gaussianity of the mixtures, pnGICA provides a competitive performance with similar results.

Fig. 11. The results of (a) SOBI, (b) Gauss-TD, and (c) pnGICA algorithm.

5.3. Experiment 3. Mixture of three images

The third experiment was implemented on three gray-scale images. Firstly, each image of size 128×128 was scanned line by line to produce the one dimensional signal with 16,384 samples. Secondly, the three signals were mixed by the linear mixing matrix which was randomly generated

$$\mathbf{A} = \begin{pmatrix} 0.8395 & 0.2756 & 0.1872 \\ -0.0745 & -0.5767 & -0.0091 \\ 0.7815 & -0.2412 & -0.9750 \end{pmatrix}. \quad (19)$$

The linear mixed signals were then undergone a nonlinear transformation to produce the nonlinear mixture $x_i(t) = f_i(v_i(t))$ using the following functions:

$$f_1(v) = \exp(2.5v), \quad f_2(v) = v^3, \quad f_3(v) = 1/(2+v). \quad (20)$$

The three original and nonlinear mixed images are shown in Fig. 12.

Since the input samples are quite large, the number of iterations in the training loop needed to be increased. The pnGICA's configuration, therefore, was set at $N_b = 8000$,

Table 3

Four speech signals experiment—correlation coefficient r between the original sources and the extracted signals. (Bold number represents correlation coefficient between a source and its corresponding estimate)

	s_1	s_2	s_3	s_4
SOBI				
y_1	0.1407	-0.4551	-0.1349	0.2594
y_2	-0.9309	-0.0544	0.0379	0.3180
y_3	0.2438	0.3680	0.1621	0.7837
y_4	0.0077	0.3824	-0.8167	0.0811
Gauss-TD				
y_1	0.9853	-0.0311	-0.009	0.0047
y_2	-0.0093	0.0032	0.0057	-0.9867
y_3	-0.0064	0.9798	-0.0019	-0.0215
y_4	0.0137	-0.0019	0.9764	0.0592
pnGICA				
y_1	0.0901	-0.0041	-0.0231	0.9813
y_2	-0.9903	0.0246	0.0264	0.0866
y_3	0.0248	-0.0407	0.9660	0.0738
y_4	0.0094	0.9386	-0.0212	0.0141

Fig. 12. Plots of (a) the three original images (Debbie, Lena, and Baboon) and (b) their post nonlinear mixtures.

$N_s = 20$, $\mu = 0.4$, and $L = 50$. The method for linear demixing was still the SOBI algorithm [5]. After separating the mixtures by pnGICA, the final output signals y_i were converted back to images and verified with the original ones. These separated images are

Fig. 13. The output images of (a) SOBI, (b) Gauss-TD, and (c) pnGICA algorithm (for a better view all the negative images were inverted).

plotted in Fig. 13. It can be seen that the pnGICA is capable of extracting the original images quite effectively. Whereas, the SOBI [5] was not successful in estimating the original images.

The correlation coefficients between output signals $y_i(t)$ and the original signals $s_i(t)$ for the SOBI, Gauss-TD, and pnGICA are provided in Table 4. Once again, pnGICA proved to be effective in separating sources from their PNL mixtures, with all the estimated output coefficient values higher than 0.96. It also exhibits a competitive performance comparing to Gauss-TD algorithm.

Out of the above three experiments, we have accomplished a number of tests with the variation of original sources, the number of sources and the different nonlinear functions. The performance of pnGICA were quite positive and it was able to linearized the nonlinear

Table 4

Experiment 3: Mixture of three images—correlation coefficient r between the original sources and the extracted signals. (Bold number represents correlation coefficient between a source and its corresponding estimate)

	s_1	s_2	s_3
SOBI			
y_1	-0.8954	-0.0196	-0.0005
y_2	-0.0839	0.0834	0.8422
y_3	0.0310	-0.8489	-0.0284
Gauss-TD			
y_1	0.9752	-0.0785	0.0339
y_2	0.0008	0.9848	0.0356
y_3	0.0279	0.0378	-0.9504
pnGICA			
y_1	-0.9932	0.0734	-0.0602
y_2	-0.0043	0.9829	0.0037
y_3	-0.0223	-0.0048	0.9661

inputs effectively. In the cases where the nonlinear function are quite complex, however, the pnGICA may produce insufficient results since the ‘fake plane’ used in pnGICA is far from a plane.

6. Conclusion

A new algorithm for post nonlinear independent component analysis has been introduced by exploiting the geometric difference between the distribution of a nonlinear mixture and a linear one. A simple linearization process transforms the post nonlinear inputs to linear mixtures without any additional assumption. Moreover, the independence between the linearization stage and linear demixing stage allows users to choose the best suitable ICA algorithm for the second stage in each application. In conclusion, pnGICA has the following advantages:

- Does not use any additional assumption about the original sources and the nonlinear mixtures.
- The linearizing process does not use the independent criterion to linearize the inputs. Hence, the first and second stages in PNL separating model do not depend on each other, i.e., we can choose any suitable linear ICA algorithm for the second stage.

We are currently working on the error problem caused by the heuristic criterion in (11). A possible solution for this issue could be an extended pnGICA with multiple updating points instead of a single midpoint. The convergence conditions and the best configuration for pnGICA are also some of the issues that need more study in the future.

Finally, geometric approach to nonlinear ICA is not constrained only to PNL model, studies of applying pnGICA to a broader nonlinear ICA model are being carried on.

Acknowledgment

The authors offer their sincere thanks to the anonymous reviewers whose positive and constructive comments helped to enhance the quality and presentation of this paper.

References

- [1] A. Cichocki, S. Amari, Adaptive Blind Signal and Image Processing, Wiley, New York, 2002.
- [2] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, Wiley, New York, 2001.
- [3] A. Hyvärinen, Blind source separation by nonstationarity of variance: A cumulant-based approach, *IEEE Trans. Neural Networks* 12 (6) (2001) 1471–1474.
- [4] D.T. Pham, J.F. Cardoso, Blind separation of instantaneous mixtures of nonstationary sources, *IEEE Trans. Signal Process.* 49 (9) (2001) 1837–1848.
- [5] A. Belouchrani, K. Abed-Meraim, J.F. Cardoso, E. Moulines, A blind source separation technique using second-order statistics, *IEEE Trans. Signal Process.* 45 (2) (1997) 434–444.
- [6] A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis, *IEEE Trans. Neural Networks* 10 (3) (1999) 626–634.
- [7] P. Pajunen, A. Hyvärinen, J. Karhunen, Nonlinear blind source separation by self-organizing maps, in: *Proc. Int. Conf. on Neural Information Processing (ICONIP '96)*, vol. 2, Springer-Verlag, 1996, pp. 1207–1210.
- [8] P. Pajunen, J. Karhunen, A maximum likelihood approach to nonlinear blind source separation, in: *Proc. Int. Conf. on Artificial Neural Networks (ICANN '97)*, 1997, pp. 541–546.
- [9] A. Hyvärinen, P. Pajunen, Nonlinear independent component analysis: existence and uniqueness results, *Neural Networks* 12 (3) (1999) 429–439.
- [10] C. Jutten, J. Karhunen, Advances in nonlinear blind source separation, in: *Proc. Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA '03)*, Nara, Japan, 2003, pp. 245–256.
- [11] M. Babaie-Zadeh, C. Jutten, K. Nayebi, A geometric approach for separating post nonlinear mixtures, in: *Proc. of the XI European Signal Processing Conf. (EUSIPCO '02)*, vol. 2, Toulouse, France, 2002, pp. 11–14.
- [12] D. Martinez, A. Bray, Nonlinear blind source separation using kernels, *IEEE Trans. Neural Networks* 14 (1) (2003) 228–235.
- [13] H. Yang, S. Amari, A. Cichocki, Information-theoretic approach to blind separation of sources in nonlinear mixture, *Signal Process.* (1998).
- [14] Y. Tan, J. Wang, J.M. Zurada, Nonlinear blind source separation using a radial basis function network, *IEEE Trans. Neural Networks* 12 (1) (2001) 124–134.
- [15] A. Taleb, C. Jutten, Source separation in post-nonlinear mixtures, *IEEE Trans. Signal Process.* 47 (10) (1999) 2807–2820.
- [16] A. Parashiv-Ionescu, C. Jutten, A. Ionescu, A. Chovet, A. Rusu, High performance magnetic field smart sensor arrays with source separation, in: *Proc. of the 1st Int. Conf. on Modeling and Simulation of Microsystems (MSM '98)*, CA, USA, 1998, pp. 666–671.
- [17] S. Prakriya, D. Hatzinakos, Blind identification of LTI-ZMNL-LTI nonlinear channel models, *IEEE Trans. Signal Process.* 43 (12) (1995) 3007–3013.
- [18] M.J. Korenberg, I.W. Hunter, The identification of nonlinear biological systems: LNL cascade models, *Biol. Cybernet.* 55 (1996) 125–134.
- [19] A. Ziehe, M. Kawanabe, S. Harmeling, K.R. Mueller, Blind separation of post-nonlinear mixtures using gaussianizing transformations and temporal decorrelation, in: *Proc. Int. Symposium on Independent Component Analysis and Blind Signal Separation (ICA '03)*, Nara, Japan, 2003, pp. 269–274.
- [20] A. Ziehe, M. Kawanabe, S. Harmeling, K.R. Mueller, Blind separation of post-nonlinear mixtures using ace and temporal decorrelation, in: *Proc. Int. Conf. on Independent Component Analysis and Blind Signal Separation (ICA '01)*, CA, USA, 2001, pp. 433–438.
- [21] T.V. Nguyen, J.C. Patra, E.-L. Ang, Blind image extraction from nonlinear mixtures using MLP-based ICA, in: *Proc. IEEE International Conference on Multimedia and Expo (ICME '03)*, vol. 1, 2003, Baltimore, USA, pp. 249–252.

- [22] T.V. Nguyen, J.C. Patra, A. Das, A geometric approach to post nonlinear mixture in blind source separation, in: Proc. IEEE International Conference on Communications Systems (ICCS '04), vol. 1, Singapore, 2004, pp. 260–264.
- [23] W. Cleveland, C. Loader, Smoothing of scatterplots, Technical report, AT&T Bell Laboratories, Murray Hill, NY, 1995.
- [24] J.H. Friedman, W. Steutzle, Smoothing of scatterplots, Technical report, Orion 3 Department of Statistics, Stanford University, 1982.
- [25] P. Craven, G. Wahba, Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation, *Numer. Math.* 31 (1979) 377–403.
- [26] A. Cichocki, S. Amari, K. Siwek, T. Tanaka, et al., Icalab toolboxes (2003), available at: <http://www.bsp.brain.riken.jp/ICALAB/>.