“Heaven’s Light is Our Guide”

Rajshahi University of Engineering & Technology

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

(CSE)

PROJECT REPORT

Report on: INDEPENDENT COMPONENT ANALYSIS (ICA)

COURSE TITLE:	SUBMITTED BY-	SUBMITTED TO-
SOFTWARE DEVELOPMENT PROJECT II COURSE NO:CSE3200 SUBMITTED DATE: 21 .03.2016	NAME: HUMAYUN AHMED ASHIK CLASS: 3RD year, EVEN semester ROLL NO: 133068 SESSION: 2013-2014	FIROZ MAHMUD Assistant Professor Department of CSE, RUET

ILLUSTRATION OF INDEPENDENT COMPONENT ANALYSIS (ICA)

What is feature?

Features are nothing but the unique signatures of the given image or unique properties that defines an image. Features are extracted in order to differentiate between the images

What is feature extraction?

Feature extraction is the process by which certain features of interest within an image are detected and represented for further processing. This approach is useful when image sizes are large and a reduced feature representation is required to quickly complete tasks such as image matching and retrieval.[1] It is a critical step in most computer vision and image processing solutions because it marks the transition from pictorial to non-pictorial (alphanumerical, usually quantitative) data representation. The resulting representation can be subsequently used as an input to a number of pattern recognition and classification techniques, which will then label, classify, or recognize the semantic contents of the image or its objects.

How features can be extracted from an image?

In the field of images, features might be raw pixels for simple problems. However, in natural images, usage of simple image pixels are not descriptive enough. Instead there are two main stream to follow. One is to use hand engineered feature extraction methods –
e.g.
* Scale invariant feature transform (SIFT),
* Vector of Locally Aggregated Descriptors- VLAD,
* Histogram of Oriented Gradients- HOG,
* GIST,
* Local Binary Pattern- LBP

and the another stream is to learn features that are discriminative in the given context
* Sparse Coding,
* Auto Encoders,
* Restricted Boltzmann Machines,
* Principal component Analysis- PCA,
* Independent Component Analysis- ICA,
* Latent Semantic Analysis- LSA

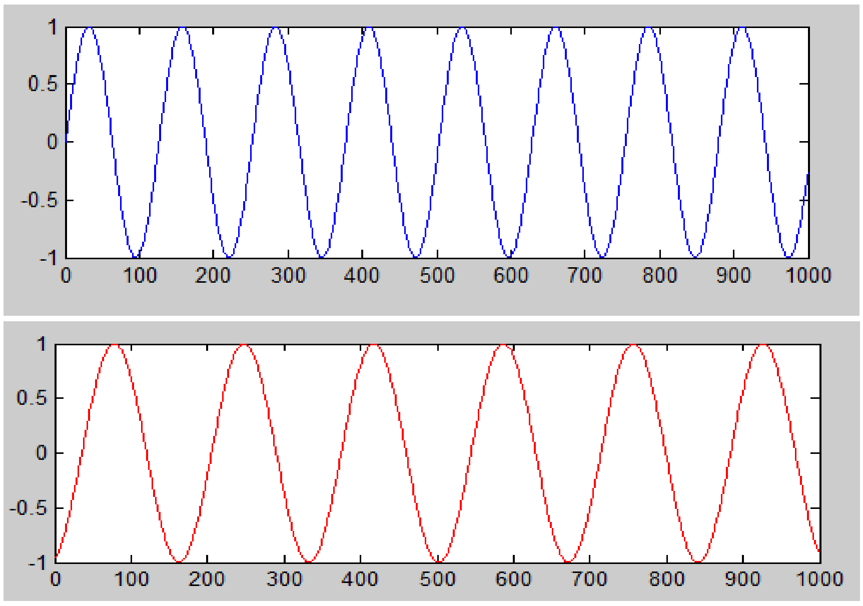
Note that second alternative, representation learning is the hot wheeled way nowadays.

What is Independent Component Analysis?

Independent component analysis (ICA) is a statistical and computational technique for revealing hidden factors that underlie sets of random variables, measurements, or signals.ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed non-Gaussian and mutually independent and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA.

Explanation with example:

In a word, ICA is a technique to separate linearly mixed sources. For instance, let’s try to mix and then separate two sources. Let’s define the time courses of 2 independent sources A (top) and B (bottom).

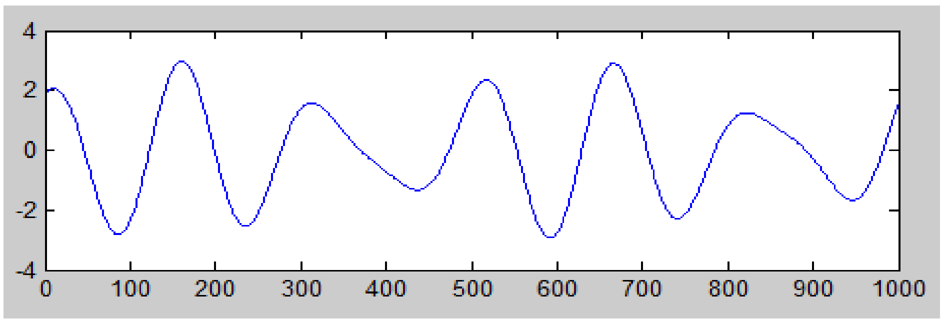


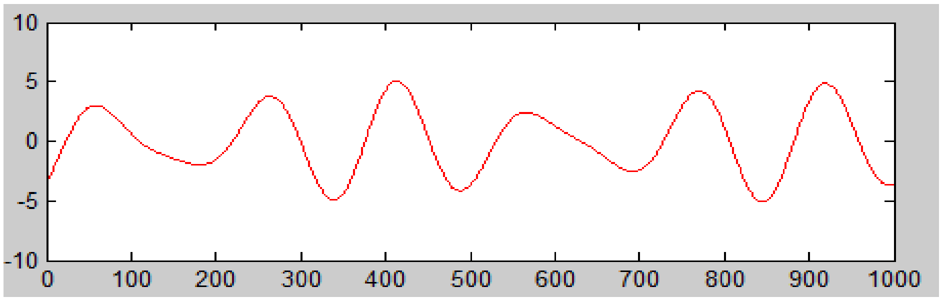
These sources are generated from the following MATLAB code:

```
A = sin(linspace(0,50, 1000));    % A
B = sin (linspace(0,37, 1000)+5); % B
figure;
subplot(2,1,1); plot(A);          % plot A
subplot(2,1,2); plot(B, 'r');     % plot B
```

Then these two sources are linearly mixed and this can be done by the following MATLAB code:

```
M1 = A - 2*B;              % mixing 1
M2 = 1.73*A+3.41*B;          % mixing 2
figure;
subplot(2,1,1); plot(M1);     % plot mixing 1
subplot(2,1,2); plot(M2, 'r'); % plot mixing 2
```



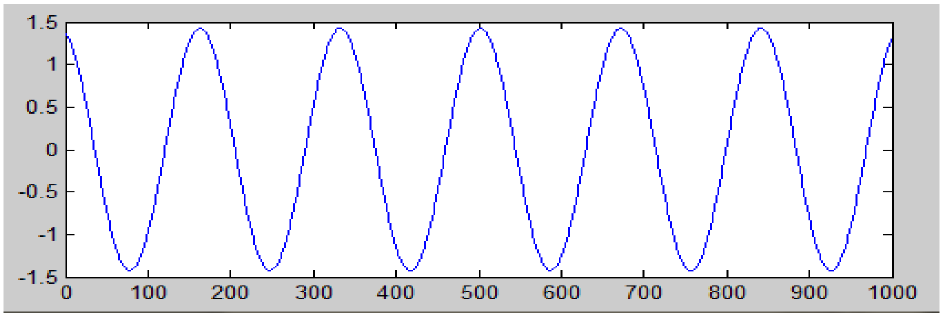


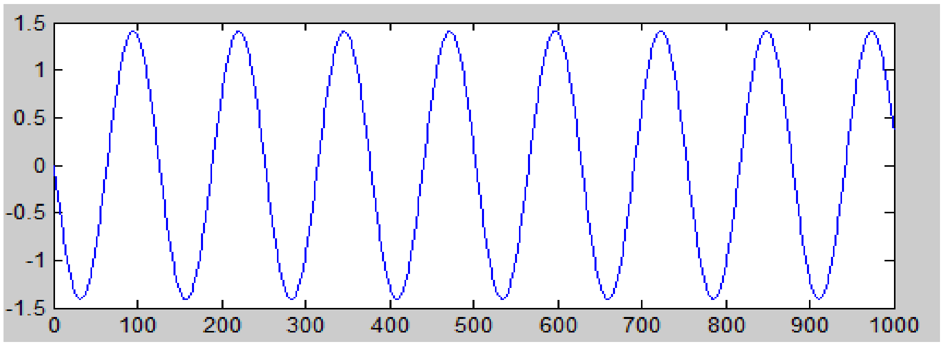
The top curve is equal to A minus twice B and the bottom the linear combination is 1.73*A +3.41*B.

We then input these two signals into the ICA algorithm (in this case, fastICA) by this following MATLAB code:

```
figure;
c = fastica([M1;M2]);          % compute and plot unminxing using fastICA
subplot(1,2,1); plot(c(1,:));
subplot(1,2,2); plot(c(2,:));
```

which is able to uncover the original activation of A and B.



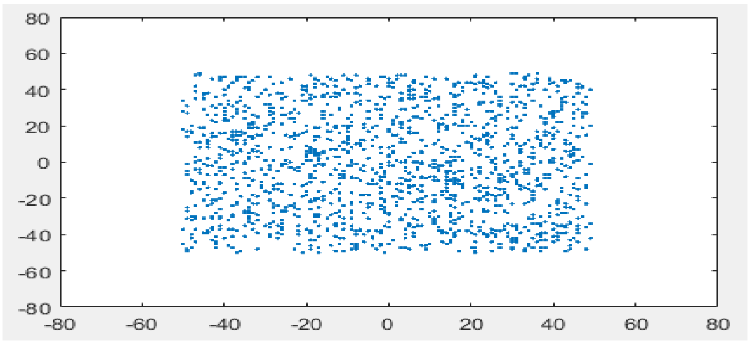


Note that the algorithm cannot recover the exact amplitude of the source activities. Note also that, in theory, ICA can only extract sources that are combined linearly.

A first step in many ICA algorithms is to whiten (or sphere) the data. This means that we remove any correlations in the data, i.e. the different channels (matrix Q) are forced to be uncorrelated.
 Once more, let's mix two random sources A and B. At each time, in the following graph, the value of A is the abscissa of the data point and the value of B is their ordinates.

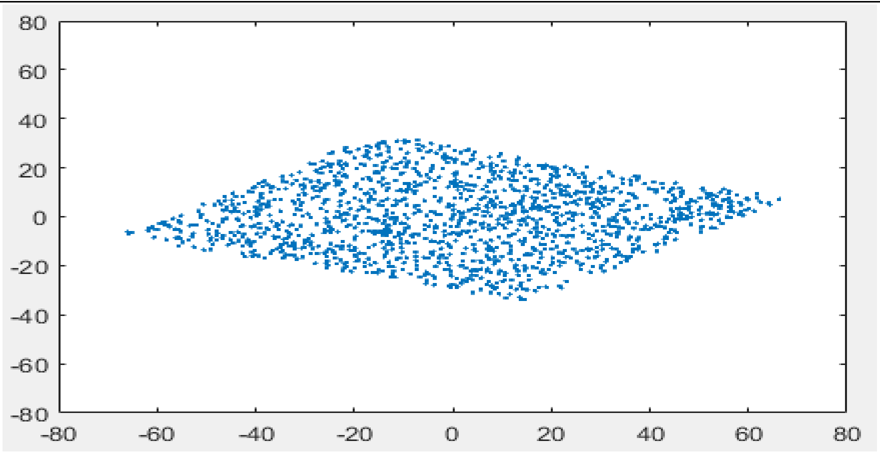
```
POINTS = 1500; % number of points to plot

% define the two random variables
% --------------------------------
for i=1:POINTS
    A(i) = round(rand*99)-50;                    % A
    B(i) = round(rand*99)-50;                    % B
end;
figure; plot(A,B, '.');                              % plot the
variables
set(gca, 'xlim', [-80 80], 'ylim', [-80 80]);
```



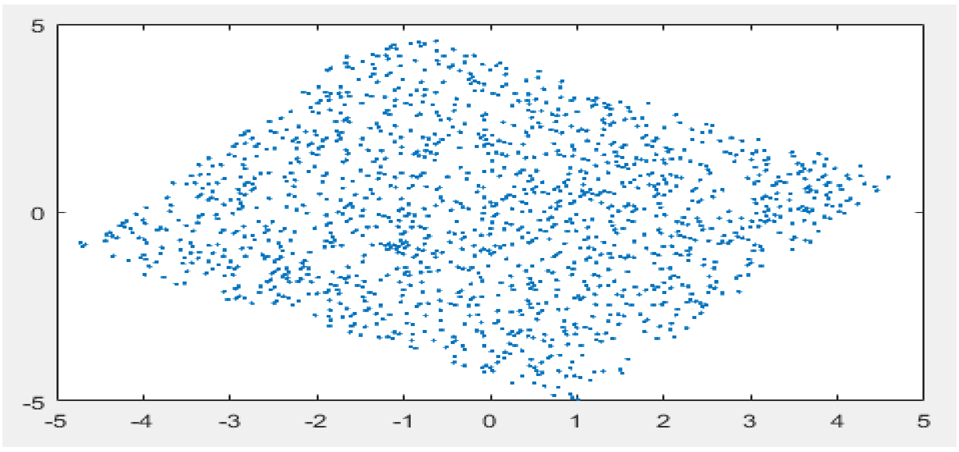
Let take two linear mixtures of A and B and plot these two new variables

```
% mix linearly these two variables
% --------------------------------
M1 = 0.54*A - 0.84*B;                                % mixing 1
M2 = 0.42*A + 0.27*B;                                % mixing 2
figure; plot(M1,M2, '.');                            % plot the
mixing
set(gca, 'ylim', get(gca, 'xlim'));                  %
redefines limits of the graph
```



Then if we whiten the two linear mixtures, we get the following plot

```
% withen the data
% ----------------
x = [M1;M2];
c=cov(x')           % covariance
sq=inv(sqrtm(c));          % inverse of square root
mx=mean(x');               % mean
xx=x-mx'*ones(1,POINTS);   % subtract the mean
xx=2*sq*xx;
cov(xx')                   % the covariance is now a
diagonal matrix
figure; plot(xx(1,:), xx(2,:), '.');
```

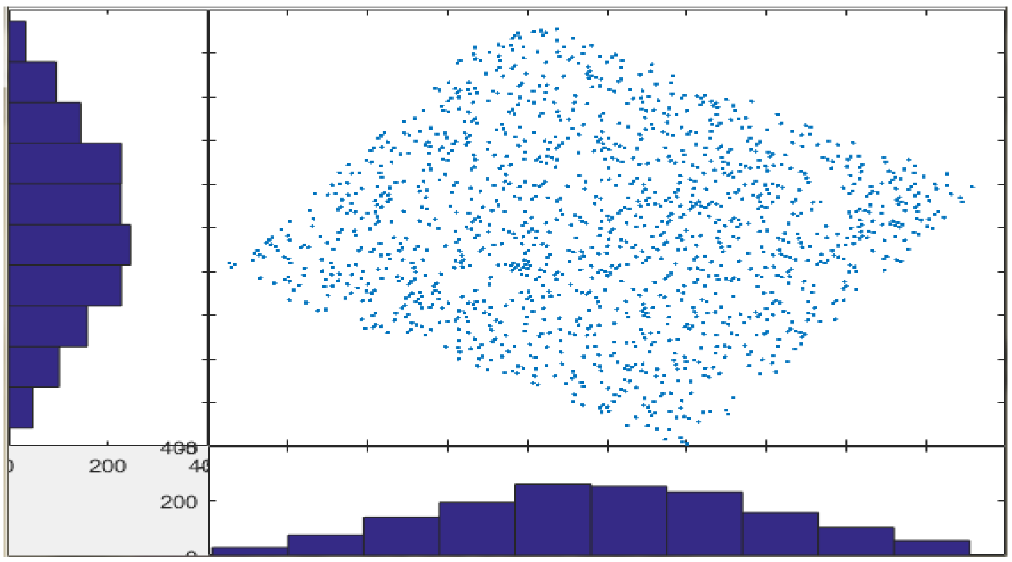


the variance on both axis is now equal and the correlation of the projection of the data on both axis is 0 (meaning that the covariance matrix is diagonal and that all the diagonal elements are equal). Then applying ICA only mean to “rotate” this representation back to the original A and B axis space.
The whitening process is simply a linear change of coordinate of the mixed data. Once the ICA solution is found in this “whitened” coordinate frame, we can easily reproject the ICA solution back into the original coordinate frame.

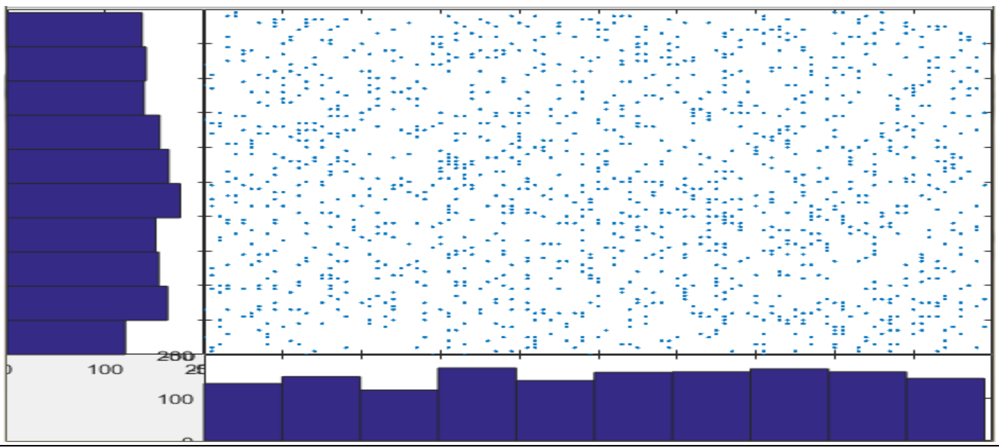
The ICA algorithm

Intuitively you can imagine that ICA rotates the whitened matrix back to the original (A,B) space (first scatter plot above). It performs the rotation by minimizing the Gaussianity of the data projected on both axes (fixed point ICA). For instance, in the example above,

```
% show projections
% ----------------
figure;
axes('position', [0.2 0.2 0.8 0.8]); plot(A,B, '.'); hold
on;
axes('position', [0   0.2 0.2 0.8]); hist(B);
set(gca, 'xdir', 'reverse'); set(gca, 'view', [90 90]);
axes('position', [0.2 0   0.8 0.2]); hist(A);
```



The projection on both axis is quite Gaussian (i.e., it looks like a bell shape curve). By contrast the projection in the original A, B space far from Gaussian



By rotating the axis and minimizing Gaussianity of the projection in the first scatter plot, ICA is able to recover the original sources which are statistically independent (this property comes from the central limit theorem which states that any linear mixture of 2 independent random variables is more Gaussian than the original variables).

PRINCIPLES OF ICA ESTIMATION

ICA was a method of performing blind signal separation that aims to recover unknown sources from a set of the observed values, in which they were mixed in an unknown manner. In the basic ICA model, the observed mixture signals x(t) can be expressed as

$$x(t) = As(t) \qquad (1)$$

Where A is an unknown mixing matrix, and s(t) represents the latent source signals which was supposed to be statistically mutually independent. The ICA model described an additive noise vector v (t) , and gave a more realistic and general ICA model in the noising case:

$$x(t) = As(t) + v(t) \qquad (2)$$

The independent components s(t) couldn’t be directly observed and the mixing coefficients A and the noise v(t) was also assumed to be unknown. If noise is negligible, only the random variables x(t) was observed and both the components s(t) and the coefficients A must be estimated using x(t) .Then, the ICA solution was obtained in an unsupervised way that found a de-mixing matrix W. The de mixing matrix W was used to transform the observed mixture signals x (t) to give the independent signals. That was:

$$s(t) = Wx(t) \qquad (3)$$

The signals s’ (t) were the close estimation of the latent source signals s (t) . If $W = A^{-1}$, then the recovered signals $s^{\wedge}$ (t) were exactly the original sources s (t). The components of $s^{\wedge}$ (t), called independent components, were required to be as mutually independent as possible. Some main functions in ICA were described below.

A. Nongaussian was independent

Intuitively talking, the central concept in assessing the ICA model is its non-gaussian nature. As a matter of the fact, without the non-gaussian nature in the ICA modeling, the estimation would not be possible at all. This can be considered the main reason for the relatively late resurrection of ICA research:

In most of the classical statistical theory, random variables were assumed to be having a Gaussian distribution, there by impeding any methods related to ICA.

B. Measures of nongaussianity

For using nongaussianity in ICA estimation, let us have (say y), a quantitative measure of nongaussianity of a random variable. To make the things simplified, let us assume that y was centered (zero-mean) and had variance equal to one.

C. Kurtosis .

The classical measure of nongaussianity was kurtosis or the fourth-order cumulant. The kurtosis of x was classically defined by

$$\mathrm{Kurt}(y) = E\{x4\} – 3(E\{x2\})^{2} \qquad (4)$$

Actually, since we assumed that y was of unit variance, the right-hand side simplifies to E{x4} – 3, showing kurtosis was basically a normalized version of the fourth moment E{x4}. For a Gaussian y, the fourth moment equals $(E\{x2\})^2$. Thus, kurtosis was equal to zero for a Gaussian random variable. For most of the non-Gaussian random variables, kurtosis was non-zero. Kurtosis could be both positive and negative. Random variables that had a negative kurtosis were called sub-gaussian, and those with positive kurtosis were called supergaussian. Typically nongaussianity was measured by the absolute value of kurtosis. The square of kurtosis could also be used. These were zero for a gaussian variable, and greater than zero for most nongaussian random variables. There were nongaussian random variables that had zero kurtosis, but they could be considered as very rare. Kurtosis, or rather its absolute value, had been widely used as a measure of nongaussianity in ICA and related fields. The main reason was its simplicity, both computational and theoretical.

D. Negentropy

A second very important measure of nongaussianity was given by negentropy. Negentropy was based on the information theoretic quantity of (differential) entropy. To obtain a measure of nongaussianity that was zero for a gaussian variable and always nonnegative, one often used a slightly modified version of the definition of differential entropy, called negentropy.

Negentropy J was defined as follows: $J(y) = H(y_{gauss}) - H(y)$ (5)

Where entropy *H* was defined for a discrete random variable *Y* as H(y) and y_{gauss} was a Gaussian random variable of the same covariance matrix as y. Due to the above-mentioned properties, negentropy was always non-negative, and it was zero if and only if y had a Gaussian distribution. Negentropy had the additional interesting property that it was invariant for invertible linear transformations. The advantage of using negentropy, or, equivalently, differential entropy, as a measure of nongaussianity was that it was well justified by statistical theory

E. Minimization of Mutual Information

Another approach for ICA estimation, inspired by information theory, was minimization of mutual information.

III. PREPROCESSING FOR ICA

Before applying an ICA algorithm, it was quite necessary and useful to work on some preprocessing on the available data. In this section, we discussed some preprocessing techniques that made the problem of ICA estimation simpler and better conditioned.

A. Centering

This was the most elementary and essential preprocessing, which was used to center the “x”, i.e. subtract its mean vector m. E{x} there by making x a zero-mean variable. It simply implied that the input signal was zero-mean as well as; it can be seen by
taking expectations on both sides of vector–matrix notation.

B. Whitening

It is additional useful preprocessing technique in ICA which is used to first whiten the detected variables. It signifies that before the application of the ICA algorithm (and after centering), we altered the detected vector x linearly so that we could obtain a new vector x which is white in its characteristic nature, i.e. its components in the image patch are uncorrelated and their variances should also be equal to unity.

C. Further preprocessing

The accomplishment of ICA for a given data set may be governed crucially by performing some application-dependent preprocessing steps. For example, if the data consists of time-signals, some band-pass filtering may be very valuable. Note that if we filter linearly the observed signals xi(t) to obtain new signals, say x*i(t), the ICA model still holds for $x^*_i(t)$, with the same mixing matrix. This can be seen as follows. The basic denoted form is by X, the matrix which contains the observations as its latent vectors x(1), ...,x(T) as its columns, and similarly for S. Then the ICA model can be expressed as:

$$X=AS \tag{6}$$

Now, time filtering of X corresponds to multiplying X from the right by a matrix, let us call it M. This gives

$$X^*=XM=ASM=AS^* \tag{7}$$

which shows that the ICA model remains still valid.

Analysis of ICA Algorithm for Separation of Mixed Images

Code implementation:

```
clear;
close all;
clc;

a = imread('1.jpg','jpg');% read image #1
aa= imresize(a,[128,128]); % resize image 'a' into 'aa' of size 128x128

aa = round((double(aa(:,:,1))+double(aa(:,:,2))+double(aa(:,:,3)))/3); % convert 'aa' image into double
type
[tx, ty] = size(aa); % assign the size of 'aa' into 'tx' and 'ty'

sz = (tx*ty);
s1 = reshape(aa,1,sz); %original signal#1

b = imread('p2.jpg','jpg');
bb = imresize(b,[128,128]);
bb = round((double(bb(:,:,1))+double(bb(:,:,2))+double(bb(:,:,3)))/3);
[tx, ty] = size(bb);

% Convert Uint8 to double array
%Ib = double(bb);
sz = (tx*ty);
s2 = reshape(bb,1,sz);%original signal#2

% View the original image
figure;
```

```
subplot(1,2,1); imagesc(aa); title('Image original 1');axis square;
subplot(1,2,2); imagesc(bb); title('Image original 2');axis square;
colormap gray;

s = [s1; s2]; % Original Signal sources

figure; plot(s1,s2, '.');                    % plot the variables
set(gca, 'xlim', [-10 200], 'ylim', [-10 200]);  % redefines limits of the graph

% define the two random variables
A = rand(2);

% Mix original sources with unknown mixing matrics 'A'
x = A * s;

x1 = reshape(x(1,:),tx,ty);
x2 = reshape(x(2,:),tx,ty);

figure; plot(x1,x2, '.');                    % plot the variables
set(gca, 'xlim', [-5 300], 'ylim', [-5 300]);  % redefines limits of the graph

% View mixed image
figure;
subplot(1,2,1); imagesc(x1); title('Mix 1');axis square;
subplot(1,2,2); imagesc(x2); title('Mix 2');axis square;
colormap gray;

% Preprocessing ica using pca
RX = cov(x');
[E LAM] = eig(RX);  % Decomposition covariance matrix
lam = diag(LAM);    % Extract eigenvalues
lam = sqrt(1./lam); % the inverse square root
LAM = diag(lam);    % Back to form the matrix
Z = LAM * E' * x;   % PCA transformation

theta(1) = 3;
theta(2) = 3;
gain = 0.1;
it = 0;
ITMAX = 100;       % Max iteration
fi = zeros(2,sz);

I = eye(2);
W = I;             % Initial value of the mixture matrix
% Estimated independent components
y = W * x;

y = whitening(y); % Apply whitening

y1 = reshape(y(1,:),tx,ty);
```

```
y2 = reshape(y(2,:),tx,ty);



figure; plot(y1,y2, '.');                    % plot the variables
set(gca, 'xlim', [-5 7], 'ylim', [-5 7]);  % redefines limits of the graph

while(it < ITMAX)
   it = it + 1;
   y = W * Z;      % ICA transformation

   for i = 1 : 2
      fi(i,:) = sign(y(i,:)) .* power(abs(y(i,:)), theta(i) - 1);
   end
   F = I - fi * y' * W;
   %figure(4);plot(y(1,:),y(2,:),'x');

end

figure;
axes('position', [0.2 0.2 0.8 0.8]); plot(y(1,:), y(2,:), '.'); hold on;
axes('position', [0   0.2 0.2 0.8]); hist(y(1,:));  set(gca, 'view', [90 90]);
axes('position', [0.2 0   0.8 0.2]); hist(y(2,:));


figure;
subplot(1,2,1); imagesc(y1); title('Image 1 estimated');axis square;
subplot(1,2,2); imagesc(y2); title('Image 2 estimated');axis square;
colormap gray;

figure;
axes('position', [0.2 0.2 0.8 0.8]); plot(s1,s2, '.'); hold on;
axes('position', [0   0.2 0.2 0.8]); hist(s2);
set(gca, 'xdir', 'reverse'); set(gca, 'view', [90 90]);
axes('position', [0.2 0   0.8 0.2]); hist(s1);
```

Whitening function:

```
function [y] = whitening(x)
     [E, D] = eig(cov(x'));
     y = E*D^(-0.5)*E' * x;
end
```

OUTPU T:

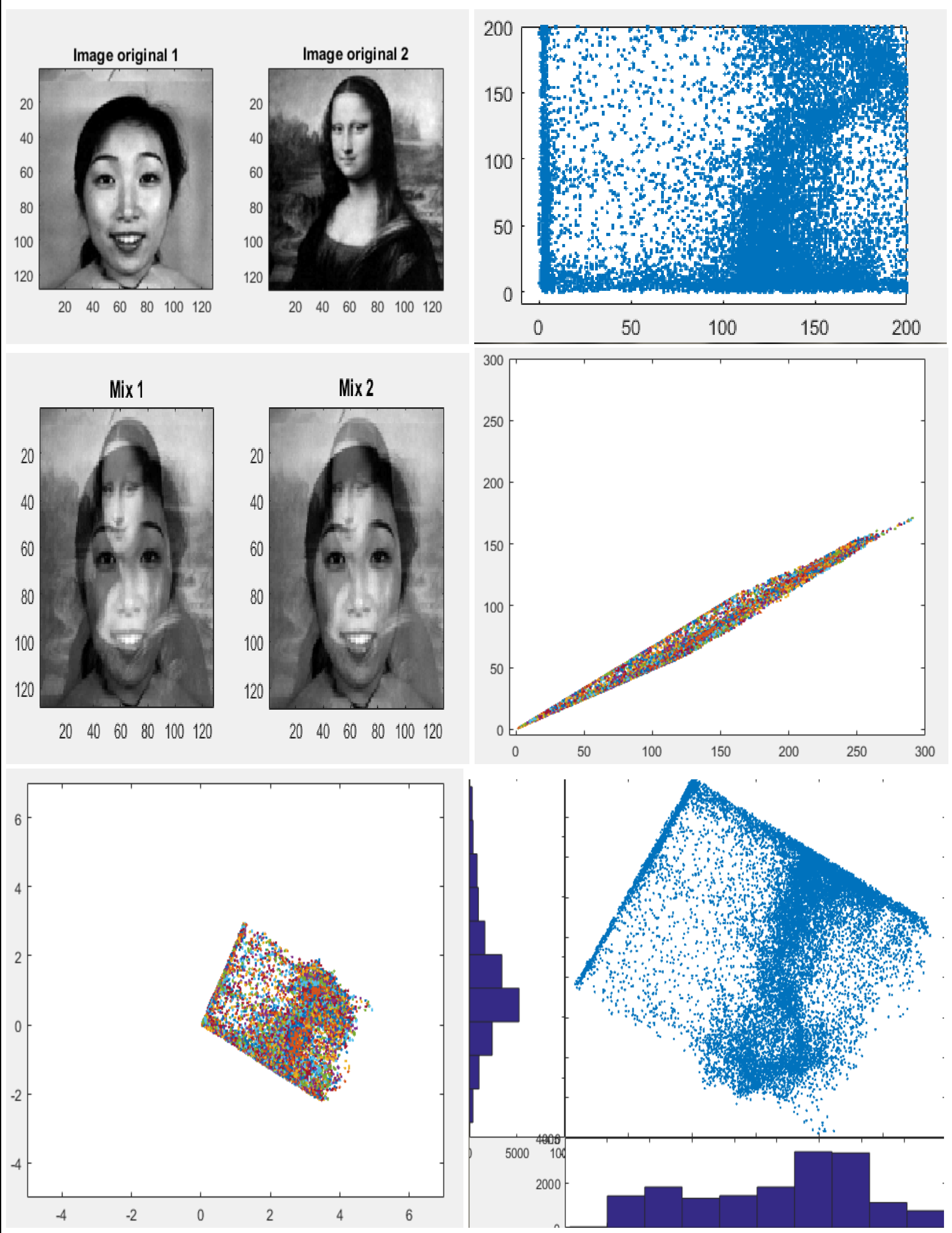
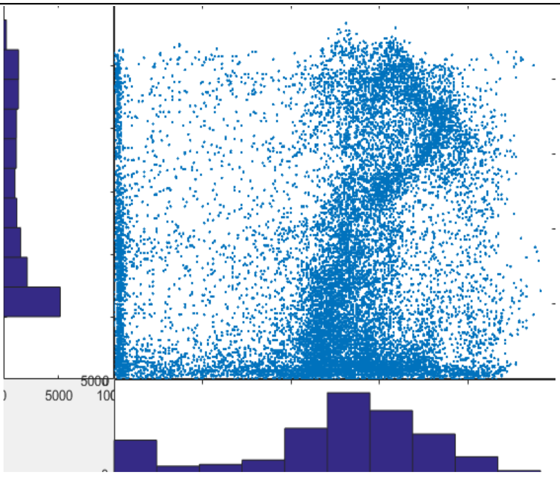
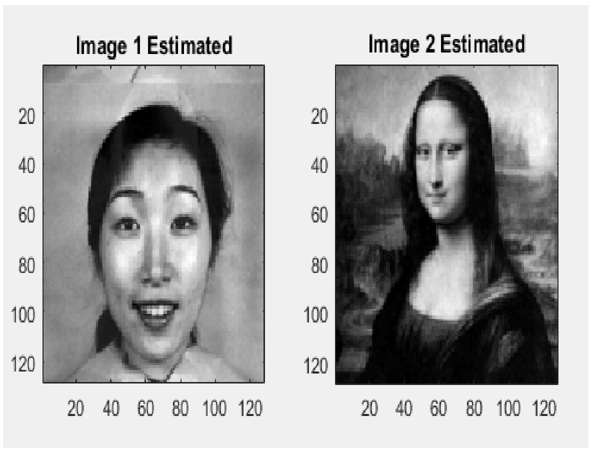


Image original 1
Image original 2
Mix 1
Mix 2





Conclusion:

Independent component analysis was performed by blind source separation technology development to a new signal processing method in recent years. It is widely used in data mining, feature extraction, neural network and taking many fields. It is to find a linear transformation for non-Gauss data. Digital image processing through the ICA, can extract independent component features that can directly reflect the essence. As the input of the classifier vector, it can not only reduce processing time, but also is very beneficial to the classifiers achieving optimal performance, reduce the matching time and error rate.