Case Studies of Independent Component Analysis

For CS383C – Numerical Analysis of Linear Algebra

Alan Oursland, Judah De Paula, Nasim Mahmood

Introduction

Our project does an Independent Component Analysis on three problems, each in a different domain. We have implemented ICA in MATLAB and Java and analyzed several problems using the software. The first problem is the standard Cocktail Party Problem. The other two problems, Robot Localization and Pinhole Camera, were devised for this project to see how ICA applies to different problems.

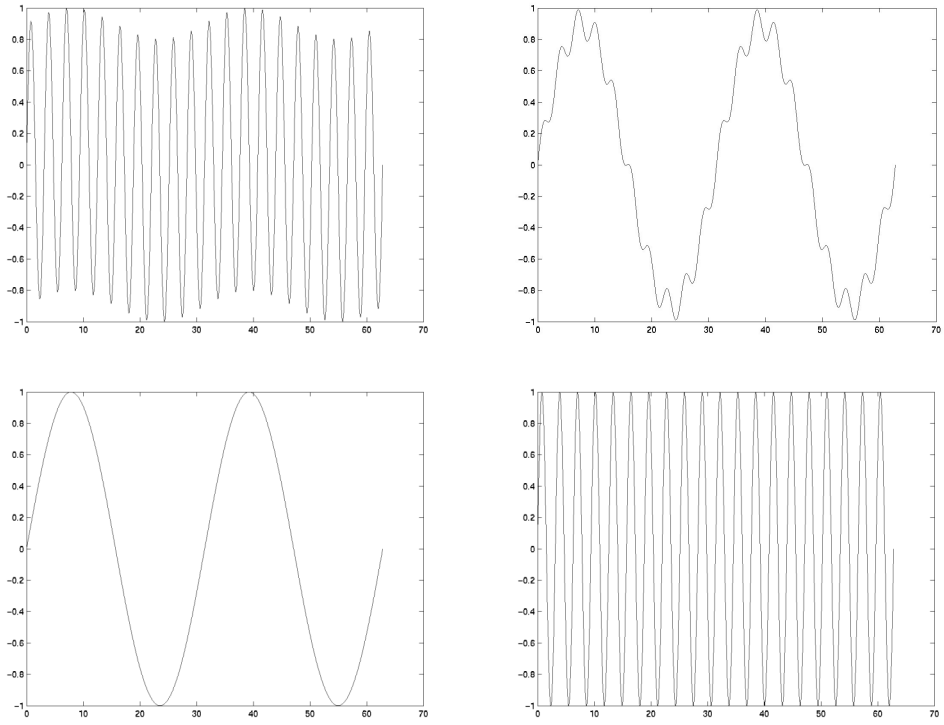
In the Cocktail Party Problem, there are several speakers, and several microphones in a room. The goal is to extract the voices of individual speakers from the different voices from the signals recorded from all of the microphones.

The second problem is robot localization, where a robot with a ring of distance sensors is placed in a square room. The goal is to determine a coordinate system for the robot with no knowledge about the organization of the sensors.

The third problem simulates a stationary one-dimensional retina covered by two slits that act as pinhole cameras. The goal is to determine the location of an object in space using only the raw retina data.

The ICA Algorithm

Independent Component Analysis is a technique that recovers a set of independent signals from a set of measured signals. It is assumed that each measured signal is a linear combination of each of the independent signals, and that there are an equal number of measured signals and independent signals. The following diagrams show an example of two independent signals and two measured signals that are linear combinations of the independent signals.



These graphs show the measured signals X1 and X2 on the top, and the independent signals S1 and S2 on the bottom. Graphs generated with the MATLAB commands:

```
x = 0:pi/100:10*pi;
s1 = sin(0.2*x);
s2 = sin(2*x);
x1 = 0.1*s1+0.9*s2;
x2 = 0.9*s1+0.1*s2;
plot(x,s1)
plot(x,s2)
plot(x,x1)
plot(x,x2)
```

We will refer to each of the original independent signals as S_i and to each of the linearly combined mixed signals a X_i. X is a column vector of n measured signals.

Each measured signal can be expressed as a linear combination of the original independent signals:

$$X_i = a_1S_1 + a_2S_2 + \ldots + a_nS_n$$

We can express the entire system of n measured signals as:

$$X = AS$$

where each row of X is a set of readings for each signal X_i; each row of S is an original signal S_i; and A is an $n \times n$ mixing matrix that generates X from S.

The goal of ICA is, given X, find S and A.

At first glance, this problem seems severely under constrained. However, ICA is looking for specific features in S that allows nearly unique solution to be found.

Constraints

We cannot recover the "real" scale of S. Any scale of the original signals can be normalized by the mixing matrix W. If S' is a set of signals with non-unit variances, then let M be a diagonal scaling matrix such that with that $a_{ii}S_i$' results in a column whose elements have variance of one.

$$X = A'S' = (A'M^{-1})(MS') = AS$$

Since the scale of the original signals cannot be recovered, we find signals with a variance of one by convention.

The scaling matrix above could also be positive or negative. For this reason, ICA produces two answers for each independent component. It does not matter which of these are selected.

There is no order to the independent signals either. Again, a permutation matrix can show that permuted columns have equivalent answers:

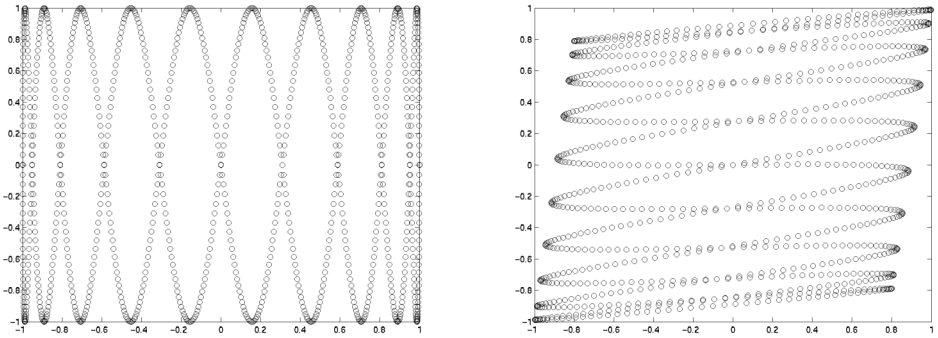
$$X = A'S' = (A'P^{-1})(PS') = AS$$

The main constraint that allows a solution to be found is that the columns of S are statistically independent. Statistical independence is described in the next section. We can think of each signal as a random variable and we want to find a matrix $W=A^{-1}$ such that WX=S such that the signals in S are maximally independent. WX=S says that the each independent signal S_i can be expressed as a linear combination of the signals in X.

Note that A must be invertible for WX=S to be valid.

Appendix A contains an introductory statistics review on statistics for the reader who needs it.

The following graphs show readings from two gaussian signals, two uniform signals, and the S and X example signals from above.



These graphs show the joint distribution between the two measured signals (X_1 and X_2) and the two measured signals (S_1 and S_2). These graphs were created with the following MATLAB commands:

```
plot(s1,s2,'o');
plot(x1,x2,'o');
```

Defining Independence

Random variables A and B are independent if the conditional probability of A with respect to A is just the probability of A. In other words, knowing a value of B tells us nothing about A. This can be expressed as the equation:

$$P(A|B) = P(A)$$

Since $P(A|B) = P(A,B)/P(B)$, where $P(A,B)$ is the joint density function of A and B.

$$P(A,B) = P(A)*P(B)$$

Another important feature of statistical independence is

$$\text{mean}(g_1(A)g_2(B)) = \text{mean}(g_1(A))\ \text{mean}(g_2(B))$$

for any functions g_1 and g_2 and $A \neq B$.

We can also look at the covariance between A and B.

$$\text{cov}(A,B) = \text{mean}(A*B) - \text{mean}(A)*\text{mean}(B)$$

For a random variable A, cov(A,A) is equal to the variance of A. If A and B are independent, then

$$\text{mean}(A*B)=\text{mean}(A)*\text{mean}(B)$$

and the covariance will be zero. The covariance of two statistically independent variables is always zero. The converse is not always true. Just because the covariance is zero does not mean A and B are independent. However, in the special case of Gaussian variables, zero covariance does imply independence. This feature of Gaussian variables is used to find columns of W in WX=S.

Previously we stated that each measured signal in X is a linear combination of the independent signals in S. The mixing matrix A is invertible such that $A^{-1}=W$. Each of the dependent components in S can also be expressed as a linear combination of the measured signals in X (S=WX).

The Central Limit Theorem states that the sum of several independent random variables, such as those in S, tends towards a Gaussian distribution. So $x_i = a_1 s_1 + a_2 s_2$ is more gaussian than either s_1 or s_2. For example, the sum of a pair of dice approximates a gaussian distribution with a mean of seven.

The Central Limit Theorem implies that if we can find a combination of the measured signals in X with minimal gaussian properties, then that signal will be one of the independent signals. Once W is determined it is a simple matter to invert it to find A.

In order to find this signal, some way to measure the nongaussianity of wX is needed. There are several ways to do this.

Estimating Independence

Kurtosis

Kurtosis is the classical method of measuring nongaussianity. When data is preprocessed to have unit variance, kurtosis is equal to the fourth moment of the data. In an intuitive sense, kurtosis measured how "spikiness" of a distribution or the size of the tails. Kurtosis is extremely simple to calculate, however, it is very sensitive to outliers in the data set. It values may be based on only a few values in the tails which means that its statistical significance is poor. Kurtosis is not robust enough for ICA.

Negentropy

The entropy of a discrete signal is equal to the sum of the products of probability of each event and the log of those probabilities. A value called differential entropy can be found for continuous function that uses the integral of the function times the log of the function. Negentropy is simply the differential entropy of a signal y, minus the differential entropy of a gaussian signal with the same covariance of y. Negentropy is always positive and is zero only if the signal is a pure gaussian signal. It is stable but difficult to calculate.

Negentropy Approximation

An approach is used to approximate negentropy in a way that is computationally less expensive than calculating negentropy directly, but still more stable than kurtosis. The following equation approximates negentropy.

$$J(x) \approx \sum_{i=1}^{p} k_i \left[mean\left(G_i(x)\right) - mean(G_i(v)) \right]^2$$

In this equation G(x) is some nonquadratic function, v is a gaussian variable with unit variance and zero mean, and k_i is some constant value. If $G(x) = x^4$ this equation becomes equal to kurtosis. There are functions that can be used for G that give a good approximation to negentropy and are less sensitive to outliers than kurtosis. Two commonly used functions are:

$$G_1(u) = \frac{1}{a_1} \log \cosh a_1 u$$

$$G_2(u) = -e^{-\frac{u^2}{2}}$$

These are called contrast functions.

Preprocessing Data

It is easier to estimate W if the measured signals have a mean of zero, a variance of one and zero correlation. The data is preprocessed to meet these goals before W is estimated.

Centering is achieved simply by subtracting the mean of signal from each reading of that signal.

A covariance matrix can be formed by taking the covariance between every pair of signals and forming a matrix. The covariance matrix will be square and symmetric. We can perform an eigenvalue decomposition on the covariance matrix and then transform the data so the covariance matrix of the transformed data is equal to the identity. This procedure is also called sphereing since it normalizes the eigenvalues of the covariance matrix.

$$C = V\Lambda V^T$$
$$x' = V\,\Lambda^{-1/2}\,V^T x$$

After whitening we will have a new mixing matrix to find. The original mixing can be recovered by applying the inverse of the whitening operation on the new mixing matrix.

$$x' = (V\,\Lambda^{-1/2}\,V^T\,A)s = A'S$$

However, now the new mixing matrix A' is orthogonal. This puts another restraint on the system and further reduces the space in which we need to search for A. Similarly, since $W'=A'^T$, the search space for W is reduced.

Finding the Mixing Matrix

A method similar to Newton's method is applied to the estimation of negentropy described above to find a mixed signal with minimum nongaussianity and maximum independence.

A Taylor series can be used to approximate a function as follows:

$$f(x_0+\varepsilon) = f(x_0) + f'(x_0)\varepsilon$$

If we are trying to find the zero of f(x) then ε can be described as the step needed from x_0 to make $f(x_0+\varepsilon)=0$.

So, $\varepsilon = \dfrac{f(x_0)}{f'(x_0)}$

Applied iteratively, we find Newton's method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

J(x) is the negentropy approximation. We want to find the minimum value of J(x) so we want to find a place where J'(x)=0. Hyvärinen has a complex analysis showing how Newton's method can be used to derive the following algorithm.

A single column of W can be found as follows:

1. select a random vector w with $||w||_2=1$
2. Let $w^+ = \text{mean}(x*G'(w^Tx)) - \text{mean}(G''(w^Tx))w$
3. repeat until w^Tw is converges to 1

The second equation is confusing and it is worth the time to show how it actually works.

- w is a mixing vector.
- w^Tx is a potential independent signal found by mixing measured signals
- $G'(w^Tx)$ and $G''(w^Tx)$ simply transform each value of the mixed signal by G' and G''.
- $x*G'(w^Tx)$ takes the inner product of each signal in X and the signal $G'(w^Tx)$. The result is a vector with a number of elements equal to the number of signals.
- $\text{mean}(x*G'(w^Tx))$ the same vector $x*G'(w^Tx)$, but each element is divided by the number of samples in each signal. This seems to be a strange definition of mean.
- $\text{mean}(G''(w^Tx))$ is the average value of all of the values of in the signal $G''(w^Tx)$ and is a scalar value.
- $\text{mean}(x*G'(w^Tx)) - \text{mean}(G''(w^Tx))w$ => vector – scalar*vector and the units work out.

We can find multiple columns of W the same way, but we need to decorrelate the rows at each step to prevent multiple columns from converging to the same solution. This is similar to orthogonalization of eigenvectors. Gram-Schmidt can be used to orthogonalize each vector with the previously discovered vectors.

Alternatively, all of the vectors can be calculated at once. The "square root" method is used to decorrelate all of the vectors at once.

$$W = (WW^T)^{-1/2}W$$

This method requires an eigenvalue decomposition and is computationally expensive.

This can also be calculated iteratively:

1. $W = W / \text{sqrt}(||WW^T||)$
2. $W = 3/2*W - ½*WW^TW$
3. Repeat step 2 until convergence

The norm is step one can be the 1-norm or 2-norm, but not the Frobenius norm.

Experiments

Cocktail Party Problem

Let there are n people in a room speaking simultaneously. We have m microphones, which we place in different locations of the room. The microphones give us m recorded time signals, which we could denote by $x_1(t)$, $x_2(t)$,… , $x_m(t)$ with x_1, x_2, … ,x_m the amplitudes, and t the time index. Each of these recorded signals is a weighted sum of the speech signals emitted by the n speakers, which we denote by $s_1(t)$, $s_2(t)$,…, $s_n(t)$. We could express this as a linear equation:

$$
\begin{aligned}
x_1(t) &= a_{11}s_1 + a_{12}s_2 + \cdots + a_{1n}s_n \\
x_2(t) &= a_{21}s_1 + a_{22}s_2 + \cdots + a_{2n}s_n \\
&\vdots \\
x_m(t) &= a_{m1}s_1 + a_{m2}s_2 + \cdots + a_{mn}s_n
\end{aligned}
$$

where a_{ij} 's are some parameters that depend on the distance of the microphones from the speakers. The a_{ij} 's are unknown and the problem is given the recorded signals $x_1(t)$, $x_2(t)$,… , $x_m(t)$ estimate the original speech signals $s_1(t)$, $s_2(t)$,…, $s_n(t)$.

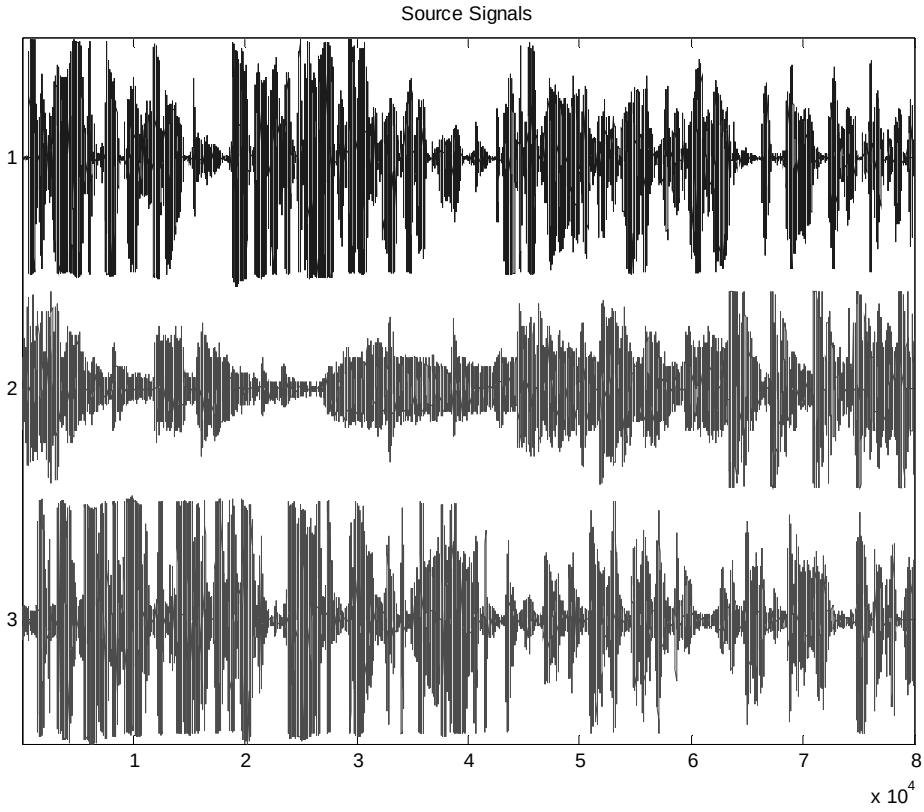
Cocktail Experiments

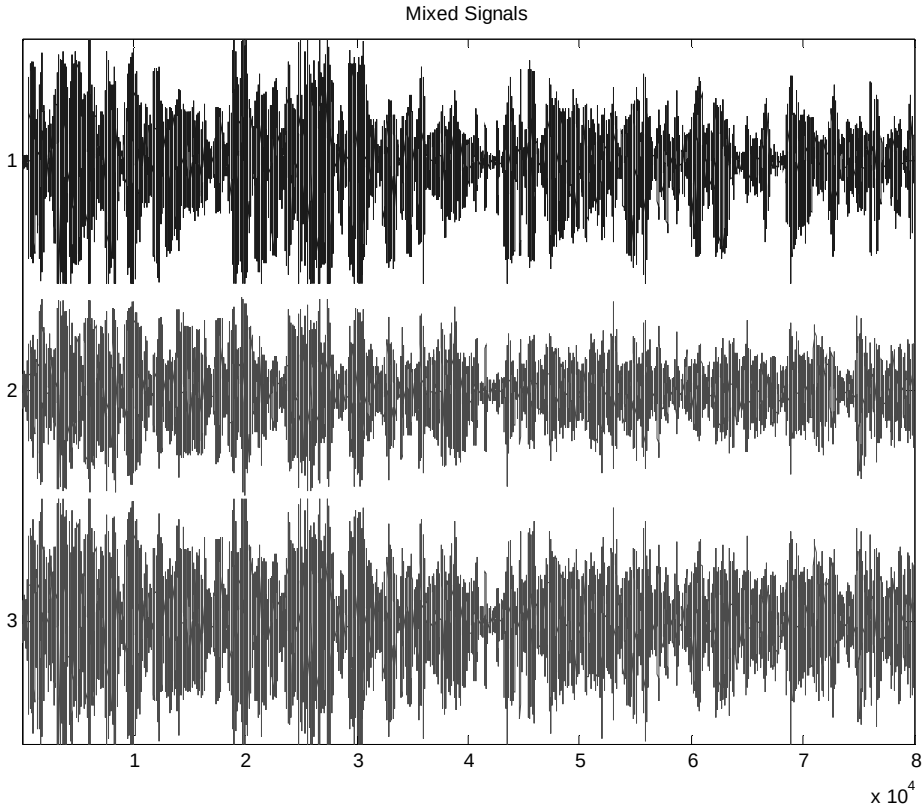
For the experiment we used three 10-second clips of wav file recorded at 8Khz sampling rate and 8 bits per sample as the original speech signals. A program was written which takes as input a collection of wav file names, mixes them randomly and then calls ICA routine on the mixed signals. For the experiment we used both the FastICA package and the ICA routine that we implemented in MATLAB. The absolute value of amplitude of the separated signals was higher than one. This is completely all right as ICA can produce signals that are multiple of the original signals. But since the wav file format requires the amplitude to be smaller than one we normalized the separated signals and then write them to wav files. The goal of this experiment is to test whether the ICA algorithm can solve the cocktail party problem. Another goal is to compare the output of the FastICA package with the ICA routine written by us.

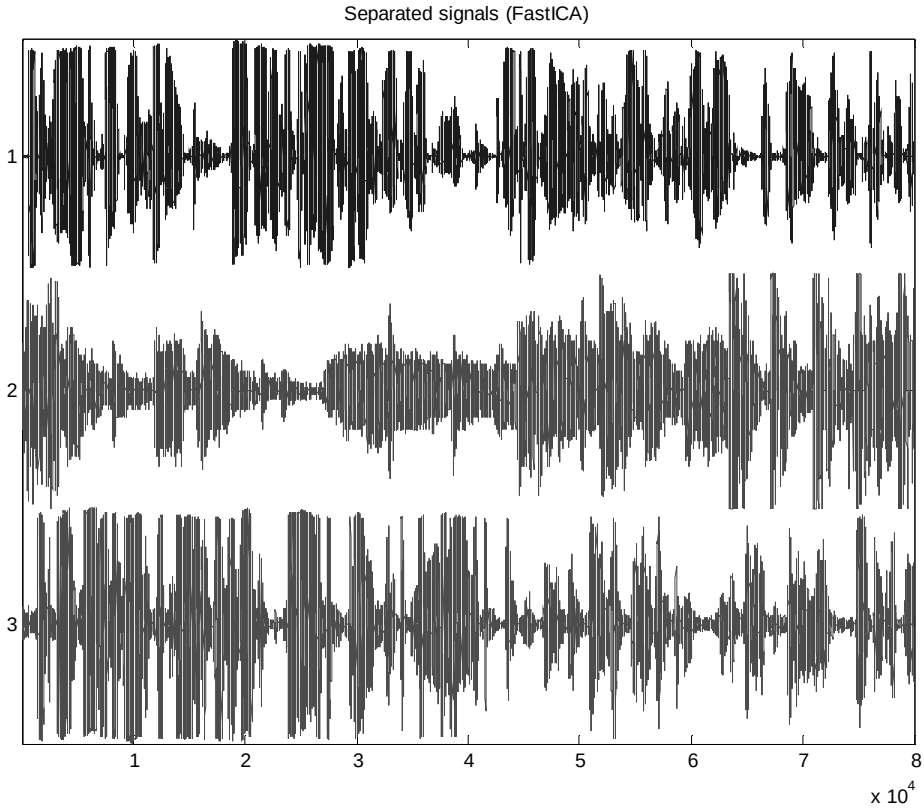
Cocktail Results

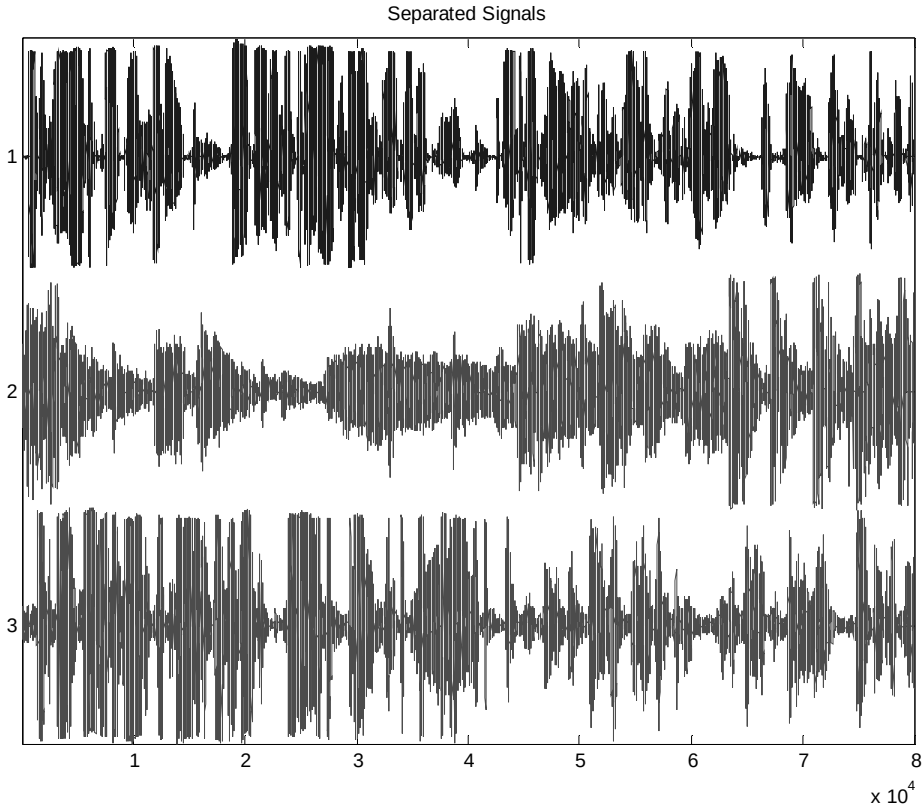
Since this experiment involves speech signals, the best way to evaluate it, is to listen to the source signals and separated signals and tell how successfully the separated signals estimate the original signals. In that respect we can say that ICA algorithm extracts the original signals from the mixed signals nearly perfectly. The output of the FastICA package is impressive. The output of our implementation is also very good.

The other way to evaluate the algorithm is to look at the waveforms of the source, mixed and separated signals. The waveforms of the source, mixed, and separated signals both by the FastICA package and our implementation is given below.


Source Signals
1
2
3
1
2
3
4
5
6
7
8
x 10^4


Mixed Signals
1
2
3
1
2
3
4
5
6
7
8
x 10^4


Separated signals (FastICA)
1
2
3
1
2
3
4
5
6
7
8
x 10^4


Separated Signals
1
2
3
1
2
3
4
5
6
7
8
x 10^4

From the waveform it is clear that the ICA algorithm successfully extracted the original signals from the mixed signals. Also the out put of our implementation and that of the FastICA is also very close.

Cocktail Experiment Conclusion

The cocktail party problem is well suited for the ICA algorithm. Also our implementation produces very good result.

Robot Localization Problem

A cylindrical robot with distance sensors around the sides is in a square room. This experiment tests if ICA can reveal a local coordinate system for a robot using data from the robot's sensors. Data was generated using a simple simulation of a robot in a square room. The robot's sensors give the distance to a wall in the direction of the sensor. The goal is to find independent components with a high correlation to the robot's actual location in the room. In lieu of a high correlation with the actual location, a different, but understandable, coordinate system is acceptable.

Robot Experiments

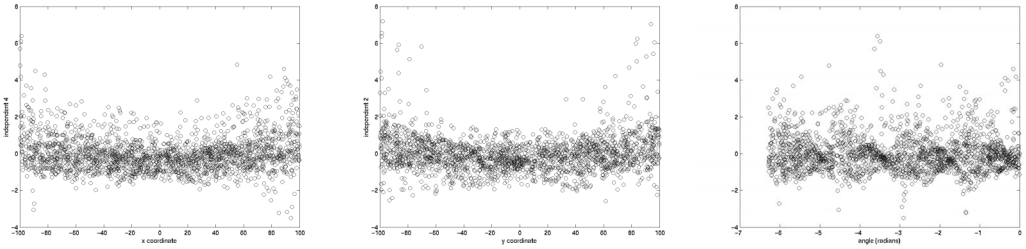
To get a good sensor distribution, the robot was placed in a random location in the room for each sensor reading. Some attempts were made to have to robot move around the room randomly, but the sample distribution did not seem to be desirable. The robot seemed to explore the middle of the room more than the edges so there was a good probability that the source data had gaussian properties.

Several experiments were run. Parameters changed for each experiment included the number of sensors on the robot and rotating the robot versus keeping it in the same direction. We were unable to interpret the results of these experiments. Another experiment was devised to determine why the results were strange.

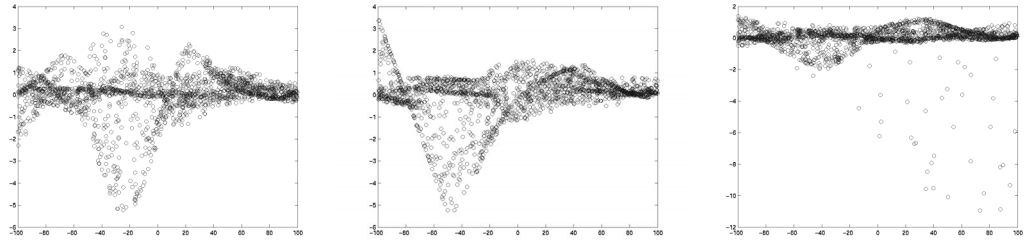
In the second set of experiments, the robot was given two distance sensors. One sensor was horizontal (at zero radians). The second sensor ranged at angles from 90 degrees to 180 degrees across 11 test runs. At ninety degrees, the sensors described the robot's coordinates and ICA found those coordinates by not changing the data. At 180 degrees, ICA produced NaN for all values. The transition from 90 to 180 shows how the independent components become less useful as the system become less suitable for linear ICA.

Robot Results

The following images are representative of the correlation between discovered independent components and the position and direction of the robot for an experiment with six sensors. There seems to be some level of correlation, but it is not clear how it would be useful. It is also interesting that the relationship with the robot direction seems to have a periodic component.



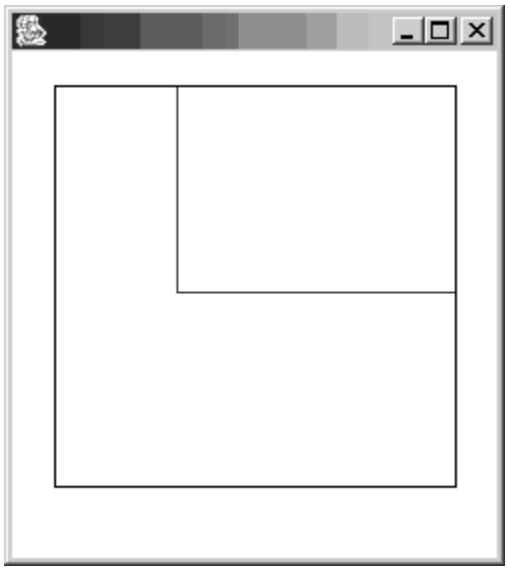
The following images were from an experiment with 31 sensors and no rotation. In this case, the graphs of the independent components versus the position seem to have damped sin and sinc functions in them.



It is not clear why these periodic signals are showing up in the relationships. The only explanation we have is that perhaps ICA is attempting to approximate the non-linearities in the system using Fourier series. However, we have nothing to back this idea up.

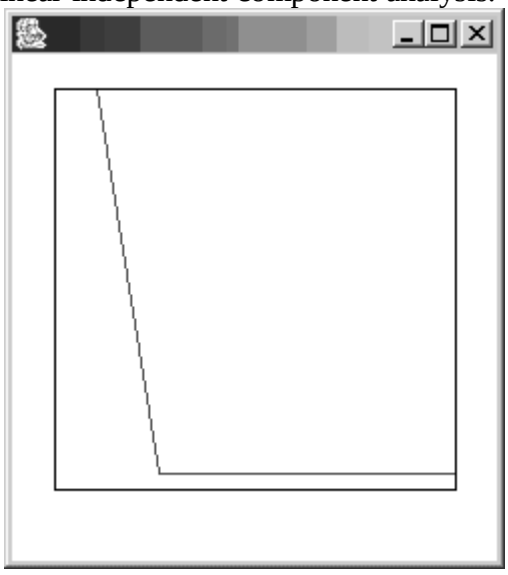
The fact that ICA did not produce signals with a high correlation to position should not be surprising. The distance sensor values are not linear combinations of the position. They are highly dependent on the positioning of the walls. We believe that the changes that occur in a sensor as it moves from one wall to an adjacent wall cause the sensor data to be non-linear and so linear ICA fails to extract meaningful data from them.

If the robot has two sensors that are perpendicular to each other, in line with the room walls, and the robot does not rotate, then the sensors provide values that can act as a coordinate system. In this case, ICA just scales the sensor values to produce the independent components.



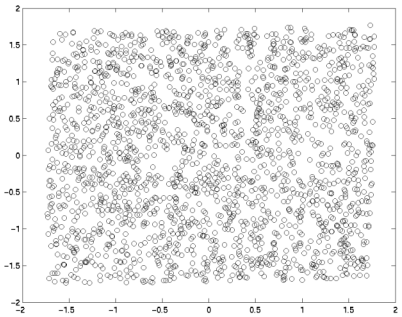
Sensor representation of a robot with 90-degree separation of the sensors.

As angle of the top sensor changes, there is a greater chance that it will hit the wall on the left instead of the top wall. The value of the sensor when it hits the top wall seems to have a non-linear relationship with the value of the sensor when it hits the left wall. This non-linearity breaks the linear independent component analysis.

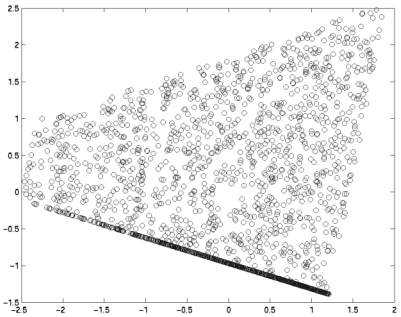


Sensor representation of a robot with 99-degree separation of the sensors.

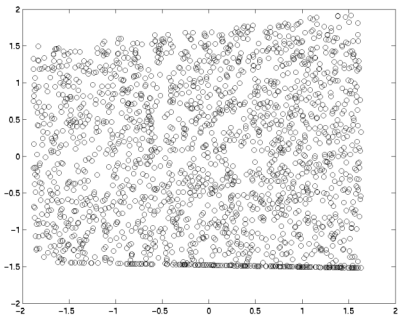
As the angle becomes close to 180 degrees, the independent components break down further. The following images show the distribution of the independent components as the sensor angle changes. The first graph matches the distribution of the robot's x and y coordinates. As the angle changes, the graph becomes less useful.



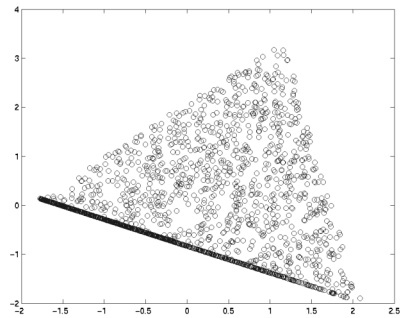
90 degrees



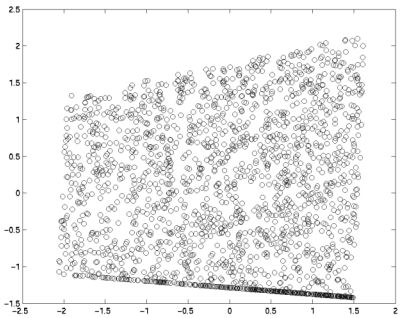
126 degrees



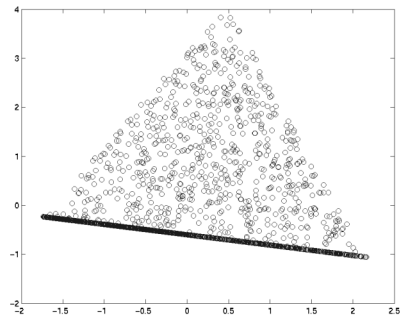
99 degrees



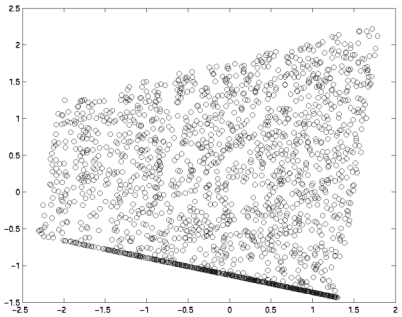
135 degrees



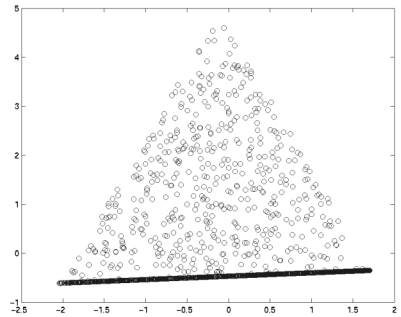
108 degrees



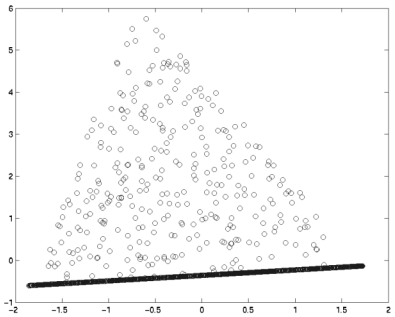
144 degrees



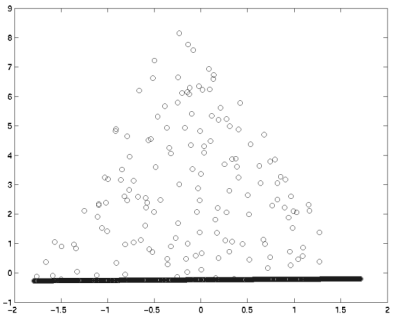
117 degrees



153 degrees



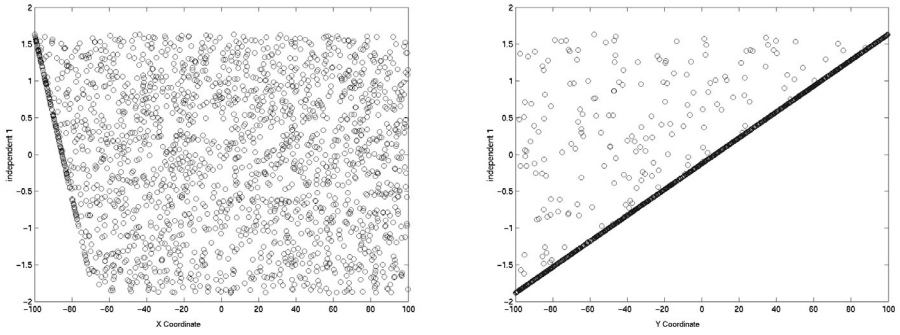
162 degrees



171 degrees

Notice that the range increases in the latter graphs. This is because the eigenvalue of the covariance matrix for the two sensors approaches zero as the sensor directions become linear. Recall that the whitening process inverts the square root of these eigenvalues and multiplies the original data by the result. This near zero vector becomes large. In fact, ICA does not produce results at all when the two angles are at 180 degrees from each other. This effect is also seen with more sensors when the robot is not rotated. This is why an odd number of sensors are used in previous experiments when the robot is not rotated.

The non-linearity is also visible by graphing the independent component against the robots positions. The following graphs show an independent component with a sensor value just off of 90 degrees against the robot's x and y coordinates.



Plots of one independent signal versus the x and y coordinates of the robot with a sensor angle of 99 degrees.

If the independent component represented a coordinate, the graph on the left would be a uniform distribution and the graph on the right would be a line. Instead, each graph is a mix of linear and uniform distributions. In the second graph, the points off of the line represent y locations where the sensor detected the left wall. The points on the line represent y location where the sensor detected the top wall.

Robot Conclusion

Overall, linear ICA does not seem to be will suited for this problem. One problem is that ICA returns the same number of independent components as the number of signals.

Dimension reducing preprocessing of the signals might help ICA produce more understandable results. ICA might be performing a Fourier estimation of non-linear components. The non-linear components are pretty evident from the correlation graphs. It might be possible to use ICA to identify subsets of the sensor space where linear descriptions are valid. In this way a space could be defined using two or more linear models.

Stereo Pinhole Retinas

We have created a model of one-dimensional pinhole retinas. For stereo effects two retinas are used. Each retina has a covering in front of the screen containing a single slit which allows light through and onto the retina. Objects move across the 2-dimensional field of vision and activate the retina using a linear projection, just as photons reflect from an object and proceed to a biological retina.

The question is: given only the activation patterns of the retinas, is it possible to extract the location of objects?

Retina Abstractions

One simplifying assumption is the object moving in front of the retina is a pinpoint ball. This causes objects to activate a single point on each of the retinas. In abstracting away size, information is lost concerning the location of the object. Normally, an object that is closer to the retina will project over a larger region of the screen. Comparison of the length of this projection can tell you which slit the object is closer to. The geometry is generally such that the total surface projection of a ball in front of one slit will be smaller than the projection against the farther slit. This is because small perturbations in position cause very large changes in the projection location when the angle of the ray is close to parallel to the screen.

Not all position information is lost in using pinpoint objects. Comparing the retina activation to the location of the slit gives adequate information: If the projection is close to the slit, the ray came in close to perpendicular, if it is far away, then the ray came in closer to the parallel.

The object size was left out to simplify the generated data. The size of the projection is dependent on the position of the object and since we are looking for the independent components, we felt it was satisfactory to leave it out in the first round. It may be that the size of the projection is information that the ICA algorithm could have used when separating the signals, but this test would require a new model of the retina.

If you recall, the proposed problem called for the use of two objects (balls.) We have used a single ball to further simplify the problem in an attempt to get better results.

The proposal also called for the use of a discrete retina, with 100 bins that were activated if the projection fell within an assigned region. This has been changed so that the retina is now continuous. This gives a much richer input signal. The use of bins may work better if we were to use non-point objects, but it is not clear what the best representation would be in such a case. We represent each time step with two real numbers, one for

each retina. A value of 0 is given to the retina when the object is directly in front of the slit. A large positive value means the object is to the right of perpendicular (with the retina to your back.) Similarly, if the value is negative, then the object is to the left of the slit.

Retina Setup

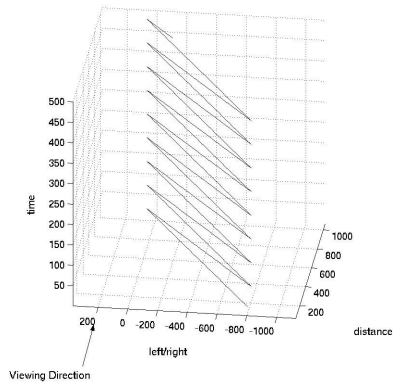
The retina is set up as follows: Both retinas are on the y-axis. The covering is 1mm in front of the retinas (1mm on the y-axis.) The slits are 35mm apart, with the right slit at 35mm (x=1,y=35) and the left slit at 85mm (x=1,y=85.) The ball is free to move within the rectangle defined by, $y \subset [-915mm,1035mm]$, $x \subset [30mm,1000mm]$. This region is selected so that the projection of the ball upon one retina will not overlap upon the other.

The ball is a point object that projects two rays. One ray passes through each slit and then projects upon the retina. Basic geometry involving symmetric triangles is used to calculate the intersection of the ray with the retina. The activated point on the retina is represented by a real integer and records the direction and distance of the activation in relation to the slit. A value of 0 means the activation is directly behind the slit. A value of 3 means that the ray hits the retina 3mm above the slit, -4 means the ray comes in 4mm below the slit.

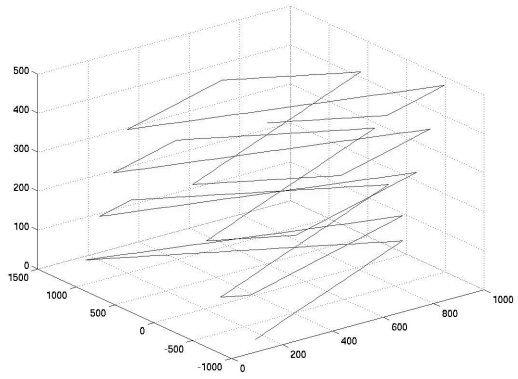
500 time steps are recorded, generating 500 pairs of real numbers. The actual X,Y location of the ball is also recorded for later comparison with the independent components calculated by the ICA algorithm.

Retina Experiment Setup

There are four primary tests. In each experiment, the retina setup remains the same, but the dynamics of the ball motion changes. For the first four trials, a ball starts at a fixed location with a set velocity. The ball then moves through space in discrete time-steps. The fourth experiment is different in that there is no sense of time. Instead each step charts the random location of a ball, which allows stochastic coverage of the entire sample space.



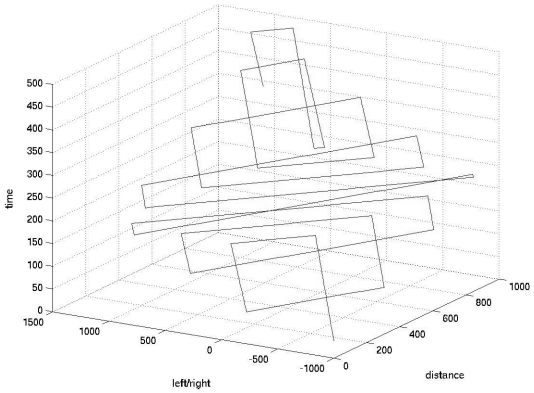
Trial 1: Planar motion of ball through time.



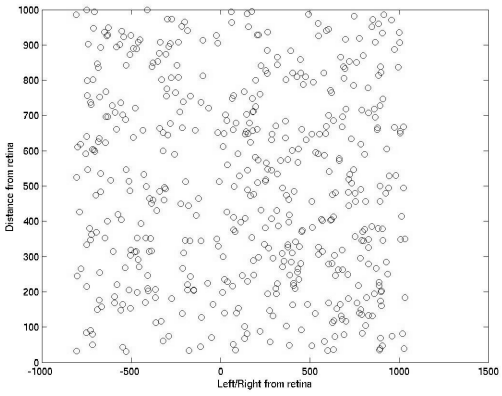
Trial 2: Planar motion of ball through time.

Trial 1: When the ball reaches a boundary, the direction is reversed in both the x and y directions. This creates a back/forth motion that covers the same path. See graph above. The motion of the ball is planar, so each time step in the graph is stacked on each the last, giving the zigzag effect. At any one time you would see a single point. Viewed from the top you would see a single diagonal line.

Trial 2: When the ball reaches a boundary wall, it bounces away from the direction of blockage, but motion continues in the open direction. This simulates a ball that is bouncing against a wall. Imagine a billiard ball bouncing around on a billiards table. See the second graph above.



Trial 3: Planar Motion of ball through time



Trial 4: Locations of random ball positions

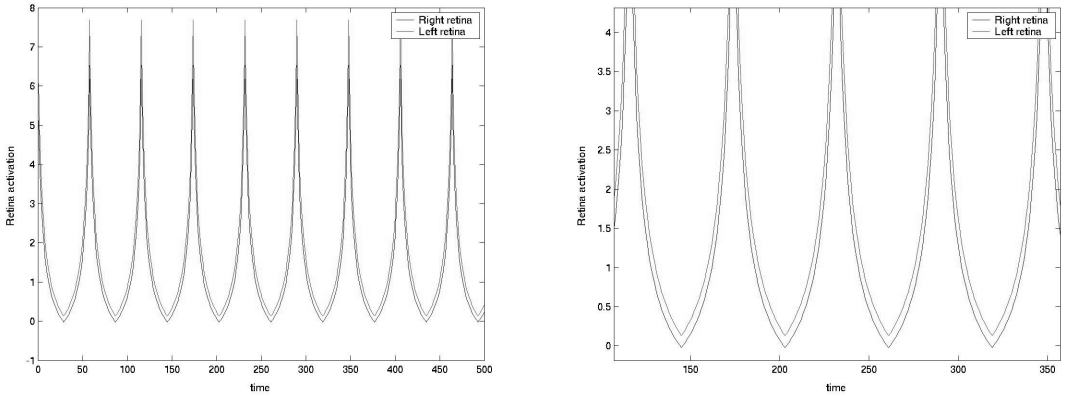
Trial 3: The bouncing is the same as in the second case, but the velocity along each axis is different. The motion along the x-axis travels at a set speed of 30mm/s, while the motion along the y-axis is moving at a different speed (50mm/s.) This introduces a non-linear relationship between the x/y coordinates.

Trial 4: Instead of tracking the motion of the ball, each step records the retinal activation of when the ball jumps to a random location. This trial is performed to see if the non-random movement of the other trials affected the results.

Retina Results

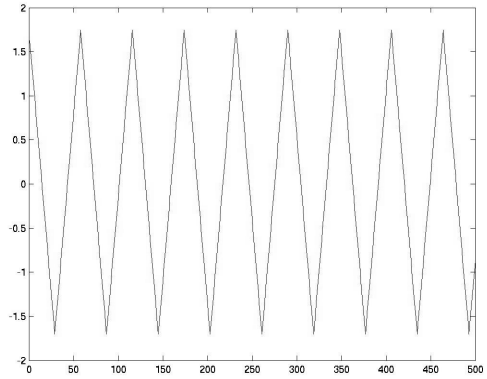
Trial 1

We recorded the following retinal activation pattern after the trial:



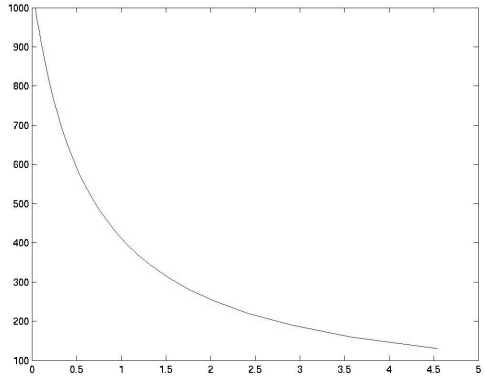
Time steps 0 to 500, with their respective activation for retinas L and R. The graph on the right is a zoom of the graph on the left. See how the overall pattern is the same except for small differences. A high activation of the retina means the object is projecting farther to the left on the retina, and the object is far to the right in space. Notice that the right retina takes smaller values than the left retina, and the left retina can go higher. This is from vision parallax.

The independent component analysis extracted the single Independent Component from the two retina signals:



Independent Component of Trial 1. x=sample(time), y=IC

When the x and y are plotted against this independent component, a curve is traced out. Here is a comparison of the distance of the ball from the retina, plotted against the computed independent component:

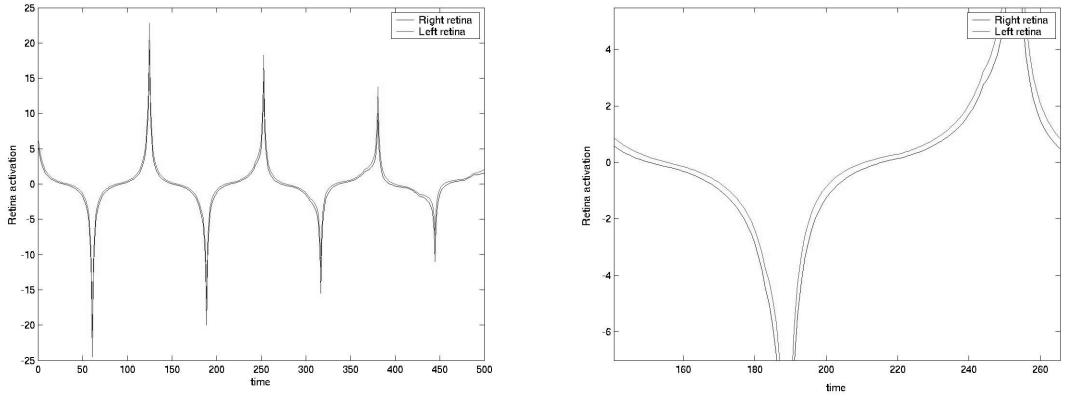


Relationship between distance and independent component. x=IC, y=distance from retina

The graph shows a monotonic relationship between the IC and distance from the retina. The graph for the left/right movement (not shown) has a similar curve. This means that given two retinal inputs, there is an ICA transformation that allows extraction of the X,Y coordinates.

Trial 2

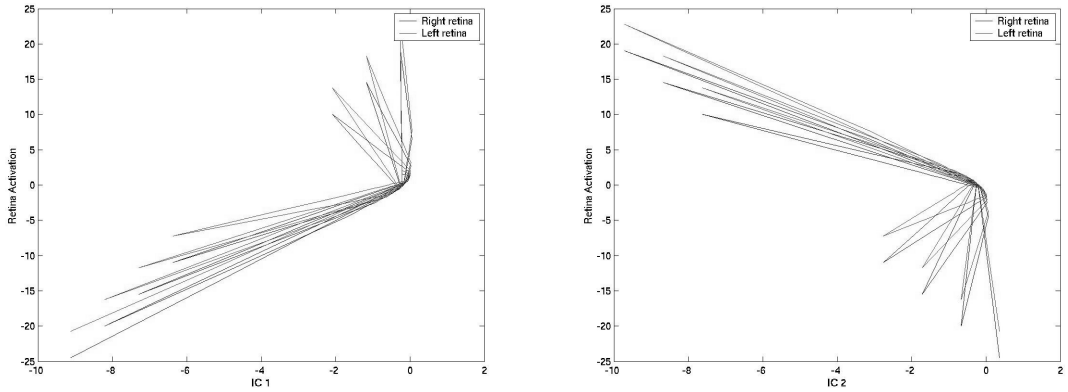
When the movement of the ball changes, so does the pattern picked up by the retinas. With billiard ball movement, the following activations are recorded:



The left graph displays the L and R retina activations superimposed over each other. The right graph is a zoom of the left. Again, the overall pattern is the same, but there are small differences due to parallax. The spikes mean that the signal on the retinas to the left and right of center move very quickly, but when the ball is in the center field of vision the speed of movement decreases.

Now that the direction of movement between x and y are no longer related to each other, the activation pattern changes. It is possible to follow the activation by comparing the activation graph to the 3D ball movement described in the Experiment Setup section. The peaks of the graph begin to fall off as the bouncing pattern changes and no longer approaches extreme x values close to the retina.

It is disappointing to find that ICA has difficulty getting an independent component in this trial. When the algorithm was run, two ICs where reported. However, when they are plotted against the original it becomes obvious that these components fail to find a linear relationship:

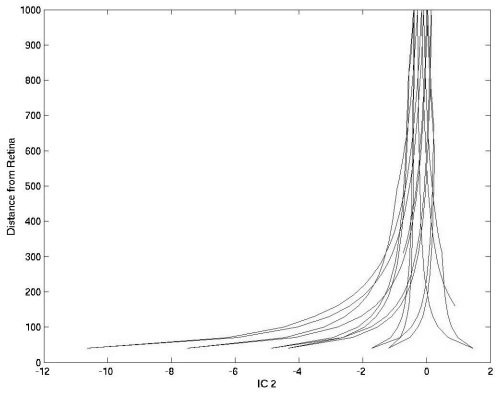


Each graph is one of the returned ICs plotted against both retinas. Though rotation has maximized linearity, it has failed to find a linear relationship.

ICA did a good job rotating the graph such that it is as linear as possible, but it is insufficient. The ICs that have been generated do allow some extraction of the data, but because a linear transformation could not be found, is not possible to definitively isolate the X,Y coordinates. The type of information that can be extracted is the same as that reported in the next trial.

Trial 3

The third trial was known to have non-linear components, so there was no surprise when ICA did not extract position information from the retina data. As in Trial 2, it is possible to extract partial information from the retina using the independent components. Graphing the data shows that for certain values of the independent components, there are constraints placed on the value of the X,Y coordinates.



Trial 3: x=IC 2, y=Distance from one retina.

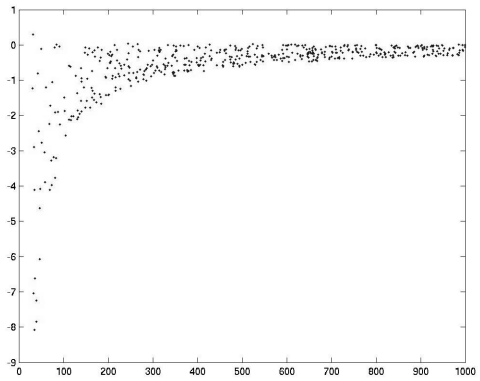
In the above example, if the IC value is close to 0, then it is not possible to discern how far away the ball is from the retina. However, as the IC moves away from 0, the value that y can take is more constrained. In other words, the closer to 0 the IC becomes, the greater the margin of error on an estimate of how far away the ball is from the retina. Our response was to see if the interaction of the two IC vectors completely constrained the location coordinates when viewed together. If it were possible to obtain a graph like above, only with the column closer to –10 on the x-axis, then it would be possible to place tight constraints upon the distance.

But this does not happen. When the IC is plotted against the other retina, the graph looks the same as above, shifted slightly, but not enough to discriminate the distance in the column around the origin. The other independent component does not give any useful information.

Given that there is a small shift between the retinas, we thought that by increasing the distance between the retina slits, we could obtain the discrimination desired. A different retina model that quadrupled the distance between the slits showed no perceptible change in the discrimination of the independent components. With the widened retinas, the columns still have a great deal of overlap and so there is still a large region of the space that is highly unconstrained.

Trial 4

After running the tests with balls that move with a set direction and speed, we wanted to see if a random distribution of points would yield better results. It could have been possible that there was an undetectable effect that hindered the other trials, so an ICA was performed on retinal data created by flashing balls randomly upon the screen. The results have the same structure as Trial 3:



x=Distance from retina, y=Independent Component. It is the same shape as in Trial 3 just the axes are reversed. Because there is no concept of time, a scatter-graph is used.

This suggests that the movement of the ball in Trial 3 is indistinguishable from a random selection of points with regard to ICA.

Retina Conclusion

The pinhole retina experiments returned mixed results. For the first trial, an independent component was found which allows X,Y coordinate discrimination. With all of the simplifying assumptions made, this trial is more of a spring motion problem than object tracking. For the more complicated examples, ICA failed because of unexpected nonlinear effects. Some information could be extracted from the independent components depending on where in the coordinate plane the object is located. More information is known than before the analysis, but it is noisy. Unrestrained vision appears to be highly nonlinear in nature, and so does not suggest itself to Independent Component Analysis.

Further experimentation is possible using different depths, retina distances, and ball sizes. But this direction of research does not warrant further pursuit, since it is possible to triangulate the location of the ball using standard geometric formulas.

Software

This section describes our implementations of ICA and the code used to generate test data. This software can be downloaded from the web at: https://webspace.utexas.edu/depaula/work/cs383c/index.html.

ICA Software

MATLAB ICA

ICA was implemented in MATLAB. The function my_ica takes as input a matrix x where each column is an observation; each row is a variable and returns the matrix s where each row is the value of the original variable. The pseudocode for the implementation is given below:

1. Center x (remove the mean from x)
2. Whiten x (uncorrelated the components)
3. for i = 1 to n
        w = random vector;
        orthogonalize initial vector w in terms of the previous components;
        normalize w;
        while (w not converged)
            w = approximation of negentropy of $w^T x$
            orthogonalize w in terms of the previous components;
            normalize w;
        end while
        W (i,:) = w;
   end for
4. s = W*whitenedx; return s;

The implementation uses deflation approach where independent components are estimated one by one.

Java ICA

ICA was implemented in Java. This code was written while attempting to understand ICA and was does not have an easily useable interface. There is no generic code that actually performs the entire ICA process. The sequence of preprocessing and performing ICA was hardcoded for each application tested.

The class ica.ICAAlgorithm implements the following features:

- ICA estimates non-gaussianity using any function that implements the interface ica.DifferentiableFunction
- A method to whiten a data set is included.
- Two techniques are implemented for finding the mixing matrix:
  1. Finding each vector of the mixing matrix independently and using Gram-Schmidt to orthogonalize.
  2. Find all of the vectors at once and using the square roots method to decorrelate vectors.
- Several basic matrix and vector operations, like multiplication, are implemented in this class.

The Java code uses the Jama library to perform the eigenvalue decomposition needed for whitening and decorrelation.

ICADistributionDemo implements the example in the ICA description section. This demo included code that can graph a two-dimensional distribution.

This demo can be run by typing the following command from the directory containing ICADistributionDemo.class: “java –cp . ICADistibutionDemo”.

Cocktail Party Experiment Software

A driver program was written in MATLAB for the cocktail party problem. There is only one function "cocktail" which takes as argument a number of wav file names. The MATLAB program then reads the wav files one b one and deals with the minimum length of all the wav files. It then mixes the signals randomly to form the input of the ICA algorithm. ICA algorithm is then invoked on the mixed signals. The returned matrix contains the separated signals. The program saves both the mixed speech signals and separated speech signals as wav files for later playback.

Robot Localization Experiment Software

The class CreateRobotData builds a data set. This file is not set up to run different experiments from the command line. The source must be edited to run different experiments.

CreateRobotData contains a Robot object with sensors, a position, and a heading. The Robot can be created to run one of two experiments. If the integer "SENSORS" is passed into the Robot constructor, the robot will have a number of sensors equals to the value of "SENSORS". If the double value "ANGLE" is passed into the constructor, the robot will have two sensors that are separated by the ANGLE amount in radians.

The static string array "filename" defines the output filename for this data. The second string defines the output filename for processed data generated by the class ProcessRobotData.

The method CreateRobotData.actionPerformed is a callback to a timer. Each time the timer is called, the robot's position is randomly changed, the sensors are updated, and the sensor values are written to a file.

The sensors values are calculated by solving four 2x2 linear systems that calculate the intersection of a sensor vector with a wall vector. The smallest positive value is used for each sensor. Complete pivoting is used to find the solution. Singular matrices are ignored since one of the four problems is guaranteed to be non-singular. The code to solve these systems is in method World.getLineIntersections.

ICA is performed on the RobotData using ProcessRobotData. This class reads the robot data file, performs the same ICA steps as ICADistributionDemo, and writes the results to a file. MATLAB was used to graph theses results.

Creating and processing data can be performed using the following commands:

```
java –cp . CreateRobotData
java –cp . ProcessRobotData
```

Stereo Pinhole Retinas Experiment Software

The model is implemented in UNIX C. The source code (named pinhole.c) can be downloaded from https://webspace.utexas.edu/depaula/work/cs383c/index.html. A Makefile is available to do the compilation.

The program generates two files: pinhole.retina.txt, and pinhole.balls.txt. Each file has 500 lines. Each line in pinhole.retina.txt has two real numbers that measure the activity in the left and right retina respectively. Each line in pinhole.balls.txt has a pair of numbers than map to the real x,y coordinates of the ball at that time step. For each of the trials, constants were changed, and the source recompiled. For the analysis phase of the experiment, a MATLAB script reads the data into matrices. The script is also available at the above website. Once the matrices are in the MATLAB environment, the ICA algorithm is run on the data, and the results plotted using MATLABs plot().

Conclusion

We learned, just as the physicists could have told us, that the world is full of nonlinear problems. At the start of the semester, our unspoken assumption was that ICA would be an excellent technique given just about any set of mixed input. Our results have shown us differently.

ICA does work well when the input is properly linear, but once something is introduced to disturb this fine balance, then all bets are off and the results become questionable. Because of the severe constraints upon ICAs usefulness, we do not see the ICA analysis becoming a magic bullet to the linear algebra community, but it will become a useful tool in the toolbox, to be called upon when a properly relevant numerical analysis is required.

The example problems that we selected had nonlinearities hidden within the domain. Robots and retinas appear to be strictly linear, as space and motion are fairly straightforward, but the relationships are much more complicated than they first appear. The cocktail problem is perfectly arranged so that the solutions practically fall out of the ICA analysis.

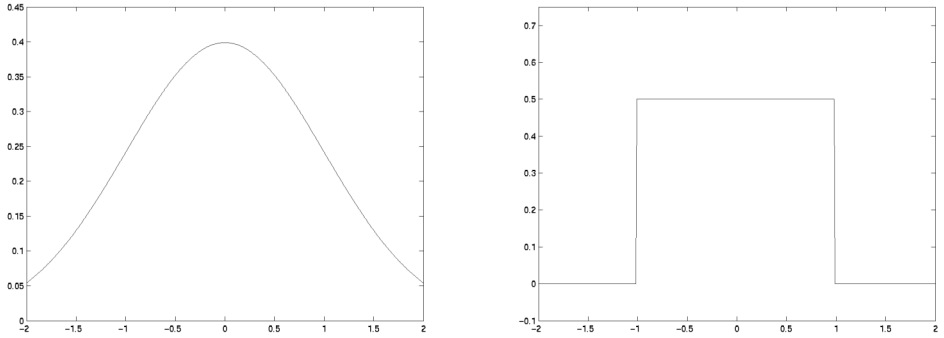
If future work were to be done with ICA, we would look into other applications using audio signal sourcing, as sound mixing has been shown to be nicely separable by the ICA algorithm.

References

1. A. Hyvärinen and E. Oja. Independent Component Analysis: Algorithms and Applications. *Neural Networks*, 13(4-5):411-430, 2000.
2. A. Hyvärinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94-128, 1999.
3. *Eric Weisstein's World of Mathematics*. Retrieved Oct-Dec 2002 from http://mathworld.wolfram.com/.

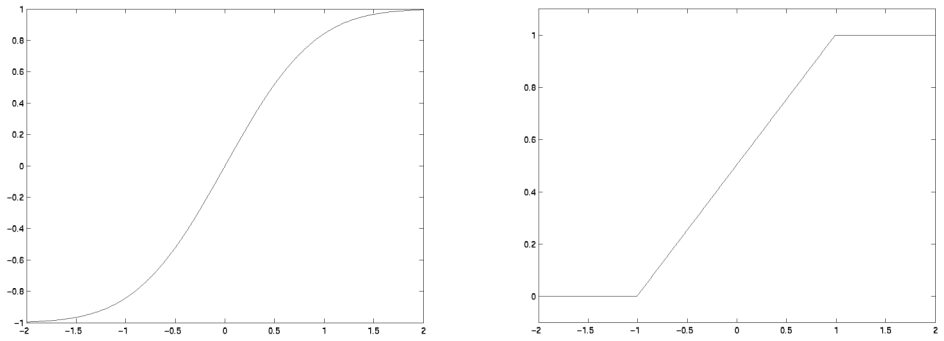
Appendix A – Statistic Review

If we have a random continuous variable, we can talk about its distribution. A random distribution describes how often a random value will appear between with an interval. The area under the distribution curve within an interval is the probability that a value will occur within that interval. The area underneath the entire curve of a distribution function must be equal to one. Two common distributions are shown below: the Gaussian density function and the uniform density function.



These graphs show the density graphs for a gaussian variable and a uniform random variable.

The distribution function of a random variable gives the chance that a random value is lower than some number. It graphs the integral of the density function from negative infinity to x. The density function is equal to the slope of the distribution function. The distribution functions for Gaussian and uniform random variables are also shown below. All distribution functions have a range from zero to one and are nondecreasing functions.



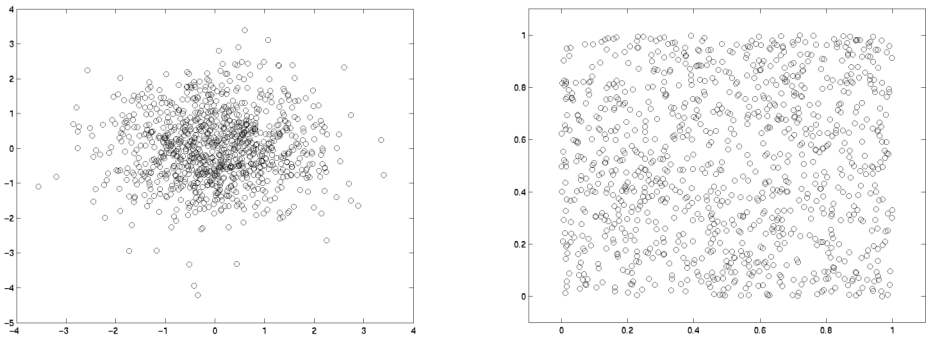
These graphs show the distribution graphs for a gaussian variable and a uniform random variable.

Random variables have an expected value – also known as the mean. The expected value is easy to calculate by taking an average of a number of samples. It can also be found by finding the point at which the distribution function is equal to 0.5.

Random variables also have a standard deviation. The standard deviation describes how far away from the mean most of the values will occur. The square of the standard deviation is called the variance. Variance is calculated by taking the mean of squares minus the squared mean.

$$\text{var}(x) = \text{mean}(x^2) - \text{mean}(x)^2$$

When we have more than one random variable we can talk about joint probability functions. The function $D(x_1,x_2)$ is a joint probability function that gives the probability that random variables $\{ X_1<x_1, X_2<x_2\}$. The volume under the joint probability density function for some range of x_1 and x_2 give the probability that both random values will occur within the range. We can get a good idea of what the joint probability density look like by graphing two sets of measure values.



These graphs show the joint distribution patterns for two gaussian variables and two uniform random variables respectively.

The graphs in this appendix were created using the following MATLAB commands:

```
x = -2:1/100:2;
u(1:100)=0;
u(100:300)=0.5;
u(300:401)=1.0;
for i=1:401 g(i) = exp(-x(i)^2/2)/sqrt(2*pi); end
up(1:100)=0;
for i=100:300 up(i) = (i-100)/200; end
up(300:401)=1;
gp = erf(x);
plot(x,g)
plot(x,u)
plot(x,gp)
plot(x,up)
rg = randn(1000,2);
ru = rand(1000,2);
plot(rg(:,1), rg(:,2), 'o');
plot(ru(:,1), ru(:,2), 'o');
```