PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MAY 2020: IN SEMESTER ASSESSMENT (ISA)

B.TECH. IV SEMESTER UE18MA251- LINEAR ALGEBRA

MINI PROJECT REPORT

ON

Cocktail Algorithm using SVD

Submitted by

1.	Varun Seshu	PES2201800074
2.	Hritik Shanbhag	PES2201800082
3.	Achyuth K Kowshik	PES2201800022

Branch & Section : 4th semester 'E' section

PROJECT EVALUATION

(For Official Use Only)

Sl.No.	Parameter	Max Marks	Marks Awarded
1	Background & Framing of the problem	4	
2	Approach and Solution	4	
3	References	4	
4	Clarity of the concepts & Creativity	4	
5	Choice of examples and understanding of the topic	4	
6	Presentation of the work	5	
	Total	25	

Name of the Course Instructor : Ms Satyavani N L

Signature of the Course Instructor : :

INDEX

1. INTRODUCTION

The ‘cocktail party party problem’ - first proposed by Colin Cherry in 1953 is his paper ‘Some Experiments on the Recognition of Speech, with One and with Two Ears’ [1] - is a psychoa-coustic phenomenon that refers to the remarkable human ability to selectively attend to and recognize one source of auditory input in a noisy environment, where the hearing interference is produced by competing speech sounds or a variety of noises that are often assumed to be independent
of each other.

There have been significant improvements on the solution to this problem over the last six decades (Bregman, 1990; Arons, 1992; Wood& Cowan, 1995; Yost, 1997; Feng & Ratnam, 2000; Bronkhorst, 2000), but it is not an understatement to say that we are not close to reaching the versatility and ease with which the human brain’s frontal, temporal and parietal lobes are able to isolate and understand speech in a plethora of noises and disturbances in the surrounding atmosphere.

In addressing the cocktail party problem, we are interested in answering a mathematical conundrum - Is it possible to build a machine capable of solving it in a satisfactory manner?

Answering the question involves the disciplines of neuroscience, cognitive psychology, and psychoacoustics, machine learning -which involves computer science- and engineering disciplines.

The three main underlying processes to solve the problem include:
1. Analysis: The analysis process mainly involves segmentation or segregation, which refers To the segmentation of an incoming auditory signal to individual channels or streams. Among the heuristics used by a listener to do the segmentation, spatial location is perhaps the most important. Specifically, sounds coming from the same location are grouped together, while sounds originating from other different directions are segregated.

2. Recognition: The recognition process involves analyzing the statistical structure of the patterns contained in a sound stream that are helpful in recognizing the patterns. The goal of recognition is to uncover the neurobiological mechanisms through which humans are able to identify a segregated sound from multiple streams with relative ease.

3. Synthesis: The synthesis process involves the reconstruction of individual sound waveforms from the separated sound streams. While synthesis is an important process carried out in the brain (Warren, 1970; Warren, Obusek, & Ackroff, 1972; Bregman, 1990), the synthesis problem is primarily of interest to the machine CPP.

However, for our implementation, we emphasise mainly on the synthesis part as we combine the matrices of the 2 different input audio streams data and process them before applying svd to the combined audio streams, then scaling down the left orthogonal matrix and unpacking its columns to get the segregated audio streams.

2. REVIEW OF LITERATURE

[1]The “Cocktail Party Problem”: What Is It? How Can It Be Solved? And Why Should Animal Behaviorists Study It?

Mark A. Bee and Christophe Micheyl
2009 Jun 8.

Animals often use acoustic signals to communicate in groups or social aggregations in which multiple individuals signal within a receiver's hearing range. Consequently, receivers face challenges related to acoustic interference and auditory masking that are not unlike the human “cocktail party problem,” which refers to the problem of perceiving speech
humans and some other animals probably face similar problems and employ similar solutions when it comes to perceiving acoustic signals in noisy social environments comprised of groups of simultaneously signaling individuals. Given the interest among animal behaviorists in acoustic signaling interactions in groups, studies of auditory scene analysis and the cocktail party problem have probably received less attention than is warranted (Hulse, 2002). We believe there are excellent reasons why animal behaviorists should study the cocktail party problem.

[2]The cocktail party problem

Josh H. McDermott
2009
Vol 19 No 22

The problem cocktail algorithm faces There are two conceptually distinct challenges for a listener in this situation
The first is the problem of sound segregation. The sounds in an auditory scene all sum together to generate the signal that enters the ear. But this mixture of sounds is itself of little use to an organism, which is typically interested in particular individual sound sources (a potential mate, for instance). The auditory system must derive the properties of individual sounds from the mixture entering the ears.

The second challenge is that of directing attention to the sound source of interest while ignoring the others, and of switching attention between sources, as when intermittently following two conversations. Most of our cognitive processes can operate only on one thing at a time, so we typically select a particular sound source on which to focus.

[3]The Cocktail Party Problem

Imon Haykin and Zhe Chen
October 2005

The cocktail party problem (CPP), first proposed by Colin Cherry[1], is a psychoacoustic phenomenon that refers to the remarkable human ability to selectively attend to and recognize one source of auditory input in a noisy environment, where the hearing interference is produced by competing speech sounds or a variety of noises that are often assumed to be independent of each other (Cherry, 1953).[1]

3. REPORT ON THE PRESENT INVESTIGATION

3.1. Singular Value Decomposition

Singular value decomposition (SVD) is a factorization of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any (m X n)

matrix via an extension of the polar decomposition..

Singular value decomposition of an (m X n) real complex matrix M is of the form UEV* where U is (m X n) real or complex unitary matrix, E is an (m X n) rectangular diagonal matrix with non negative real numbers on the diagonal and V is an (n X n) real or complex unitary matrix.

The diagonal entries of E are known as singular values of matrix M.

The number of non-singular values is equal to the rank M.

The columns of U and the columns of V are called left singular vectors and right singular vectors

Of matrix M.

Mathematical applications of the SVD include computing the pseudoinverse, matrix approximation, and determining the rank, range, and null space of a matrix. The SVD is also extremely useful in all areas of science, engineering, and statistics, such as signal processing, least squares fitting of data, and process control.

Following are some of the important properties of SVD, when the given matrix is M.[5]

1. M M(T) = L (D^2) L(T) and hence L diagonalizes the matrix M and column vectors of L are

the eigenvectors of M M(T) where D is Diagonal Matrix and L is order of m*m and R is

order of n*n and M(T) L(T) and are the transpose of matrix M,L.

2.. Similarly, M(T)M = R (D^2) R(T), R diagonalizes M and hence eigenvectors of M(T)M are column vectors of R.

3.. The singular values of matrix M i.e. σ1, σ2, σ3.....,σN are unique, however the matrices L and R are not unique.

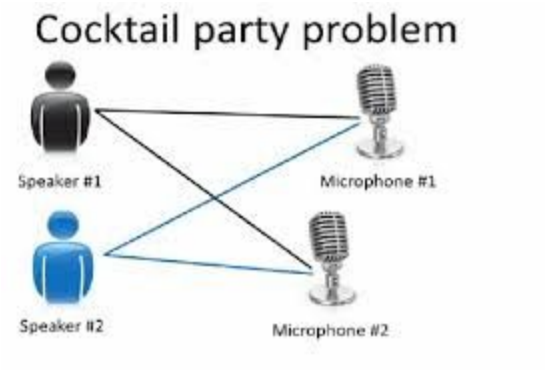
3.2 COCKTAIL PARTY PROBLEM

One of our most important faculties is our ability to listen to, and follow, one speaker in the presence of others. This is such a common experience that we may take it for granted; we may call it "the cocktail party problem." No machine has been constructed to do just this, to filter out one conversation from a number jumbled together.
—Colin Cherry, 1957.

The essence of the cocktail party problem can be given as follows: " Can we isolate and segregate the audio of a voice from a linear combination of multiple voices and sound disturbances? "

In our task to imitate this human process with a machine, we place an experiment -

With two speakers in a room and two microphones, we look at a subset of this problem where there are equal speakers and microphones, as demonstrated in the picture below.



As is evident, each microphone records a different linear combination of audio from both the Speakers. Now armed with our audio data, our problem now dissolves to separation of the voices of the two speakers from the combined inputs of both the microphones.

3.3 COCKTAIL PARTY ALGORITHM AND PROPOSED SOLUTIONS

For the cocktail party problem, the algorithm for the solution begins with appropriate pre-processing of the data.

The audio collected from the spatially separated microphones, with the same sampling rates are made into two one-dimensional matrices. These are then combined to form a 2XN matrix, where N is the sampling_rate*length_of_recording. Let this be our base matrix m.

To achieve the transformation matrix to multiply with the base matrix m, we take the covariance of m, the inverse of the resultant matrix, the square root of every element of the inverted matrix. Thus we achieve our transformation T.[4]

```
T = square_root_each_element(inverse(covariance(m)))
```

We store the row wise mean of matrix m in mean_m.

We get the processed audio from the following equation:

Processed_audio = T * (m_transpose - mean_m)

In a slide within the introductory lecture on machine learning by Stanford's Andrew Ng at Coursera, he gives the following one line solution to the cocktail party problem in the Octave programming language, given the audio sources are recorded by two spatially separated microphones:[6]

```
[W,s,v]=svd((repmat(sum(x.*x,1),size(x,1),1).*x)*x');[7]
```

Where W is the left orthogonal matrix from the singular value decomposition(U), s is sigma and v stands for transpose of the right orthogonal matrix(V transpose).

We then get the final resultant and separated audio from this equation

```
Resultant = W*Processed audio
```

Applying these transformations provides us a matrix of processed audio, with considerable amount of separation in voice.

3.4 IMPLEMENTATION OF COCKTAIL PARTY ALGORITHM

3.4.1 Taking the input from 2 audio recording

```
# Taking input from the 2 audio recording

# m, fs -> m is the audio matrix, fs is sampling rate
# fs1  and fs2 should be the same
[m1, Fs1] = audioread('mixed1.wav');
disp("mixed1.wav has been read.")

[m2, Fs2] = audioread('mixed2.wav');
disp("mixed2.wav has been read.")

# Here N is the total no of rows

disp("size of m1");
disp(size(m1));       # N X 1

disp("size of m2");
disp(size(m2));       # N X 1

m = [m1, m2];        # Here we creating matrix m from
                     # matrices m1 and m2 column-wise
disp("size of m");
disp(size(m));        # N X 2
```

3.4.2 Computing SVD

3.4.2.1 Processing audio

```
# cov(x,y) is covariance between ith variable in x and jth variable in y
# cov (x) = 1/N-1 * SUM_i (x(i) - mean(x)) * (y(i) - mean(y))
cov_m = cov(m);
disp("cov_m size ");
disp(cov_m);                                # 2 X 2

# inv(A) computes the inverse of the matrix A
inv_cov_m = inv(cov_m);
disp("inv_cov_m size ");
disp(inv_cov_m);                            # 2 X 2

# sqrtm(A) computes the matrix square root of square matrix A
sqrt_inv_cov_m = sqrtm(inv_cov_m);
disp("sqrt_inv_cov_m size ");
disp(sqrt_inv_cov_m);                       # 2 X 2

# if x=1 mean(A,x) calculates the mean of columns of A
# if x=2 mean(A,x) calculates the mean of rows of A
row_wise_mean_xx = mean(m',2);
disp("row_wise_mean_xx size ");
disp(row_wise_mean_xx);                     # 2 X 1

# Turn row_wise_mean_xx to a 2 X N
# repmat(K,m,n) return a matrix of size m X n where all the matrix co-efficients are K
mean_mat = repmat(mean(m',2), 1, size(m',2));
disp("mean_mat size ");
disp(size(mean_mat));                       # 2 X N

# processed audio
p_audio = sqrt_inv_cov_m * (m' - mean_mat); # (2 X 2) * (2 X N) = (2 X N)
disp("size of p_audio");
disp(size(p_audio));                        # 2 X N
```

3.4.2.2 implementing svd and scaling the output

```
p_audio_sq = p_audio .* p_audio;
disp("size of p_audio_sq");
disp(size(p_audio_sq));                    # 2 X N

# row wise sum
sum_p_audio_sq = sum(p_audio .* p_audio,1);
disp("size of sum_p_audio_sq");
disp(size(sum_p_audio_sq));                # 1 X N

# broadcasting the previous matrix
resize_sum_p_audio_sq = repmat(sum_p_audio_sq,size(p_audio,1),1);
disp("size of resize_sum_p_audio_sq");
disp(size(resize_sum_p_audio_sq));         # 2 X N

# [U,S,V]=svd(A,econ) returns an economy-sized decomposition,elimination any
# unnecessary rows or columns in U or V
# such that A=U*S*V
[U,s,v_T] = svd((resize_sum_p_audio_sq.*p_audio)*p_audio');

resultant = U * p_audio;

# Scale down by an empiric value
output = (resultant .* 0.1)';
```

3.4.2.3 Writing output to audio files

```
# the output audios are  stored in separated1.wav and separated2.wav

# audiowrite(filename,y,fs) writes audio data from the matrix y to
# filename at sampling rate fs with the file format determined by the file extension
audiowrite('separated1.wav', output(:, 1), Fs1);
audiowrite('separated2.wav', output(:, 2), Fs1);
```

3.5 FINAL OUTPUT

After we run the algorithm, with the 2 audio sources from microphones given as input, we produce 2 audio files with the separated audio of the 2 Speakers.

For a further sanity check of the working of our program, we have documented and checked the size of the matrices produced during the formulation of the resultant.

Given below is an explanation of the line by line output:

```
mixed1.wav has been read.
mixed2.wav has been read.
```

This is simply an indication of the microphone audio being read.

Sizes of the individual audio recordings and the size of the combined matrices:

```
size of m1
   118976        1
size of m2
   118976        1
size of m
   118976        2
```

covariance matrix of the audio recordings:

inverse of the covariance matrix:

```
inv_cov_m size
   470.00  -359.10
  -359.10   478.11
```

Square root of every element of the above matrix:

```
sqrt_inv_cov_m size
   19.6902   -9.0716
   -9.0716   19.8950
```

Row wise mean of matrix transpose:

```
row_wise_mean_xx size
  -0.000039128
  -0.000041452
```

Size of the original audio recordings subtracted by mean matrix given above:

```
mean_mat size
        2   118976
```

Size of processed audio:

```
size of p_audio
        2   118976
```

After squaring all elements in processed audio:

```
size of p_audio_sq
        2   118976
```

After performing matrix addition:

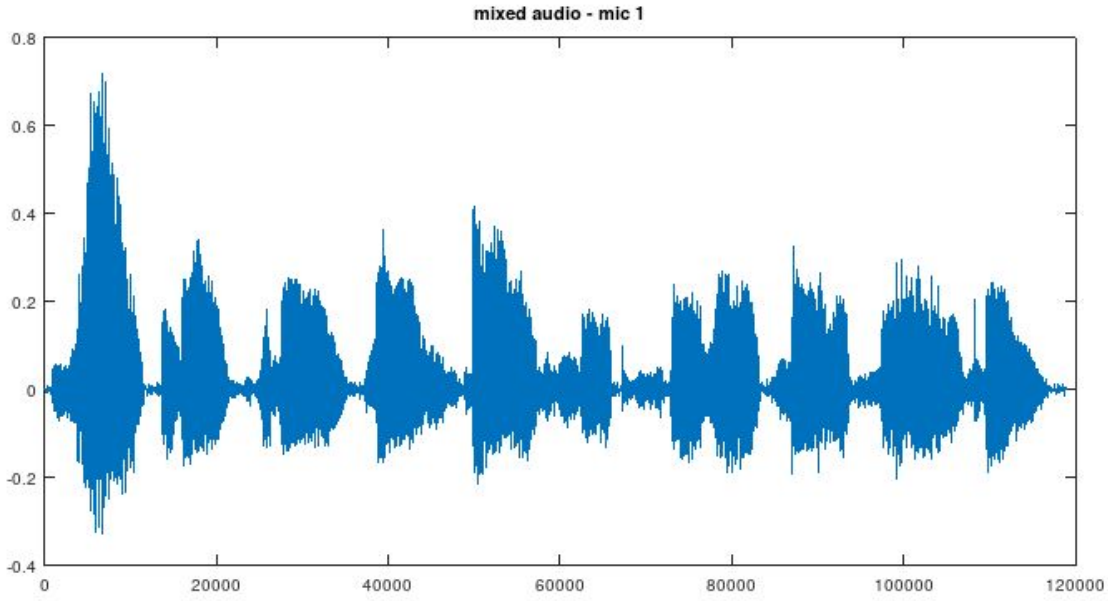
```
size of sum_p_audio_sq
        1   118976
```

After final resizing of the processed audio:

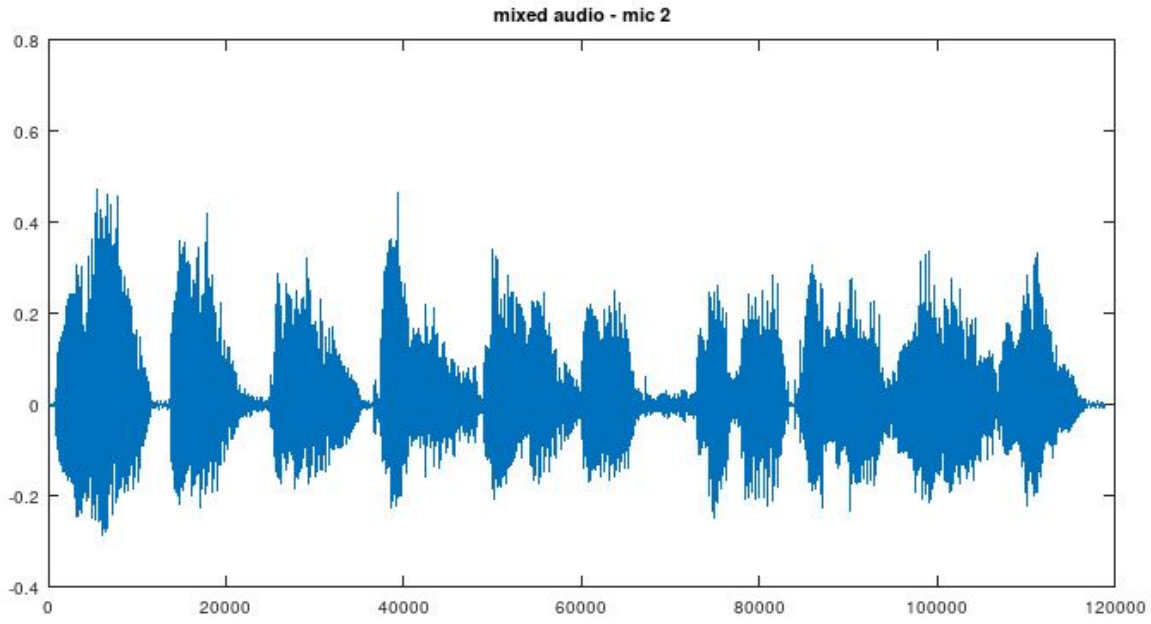
We get 2 separate audio files after separation of the 2 rows of the matrix.
Separated1.wav and Separated2.wav

4. RESULTS AND DISCUSSIONS

The raw audio recorded by microphone 1:[2]

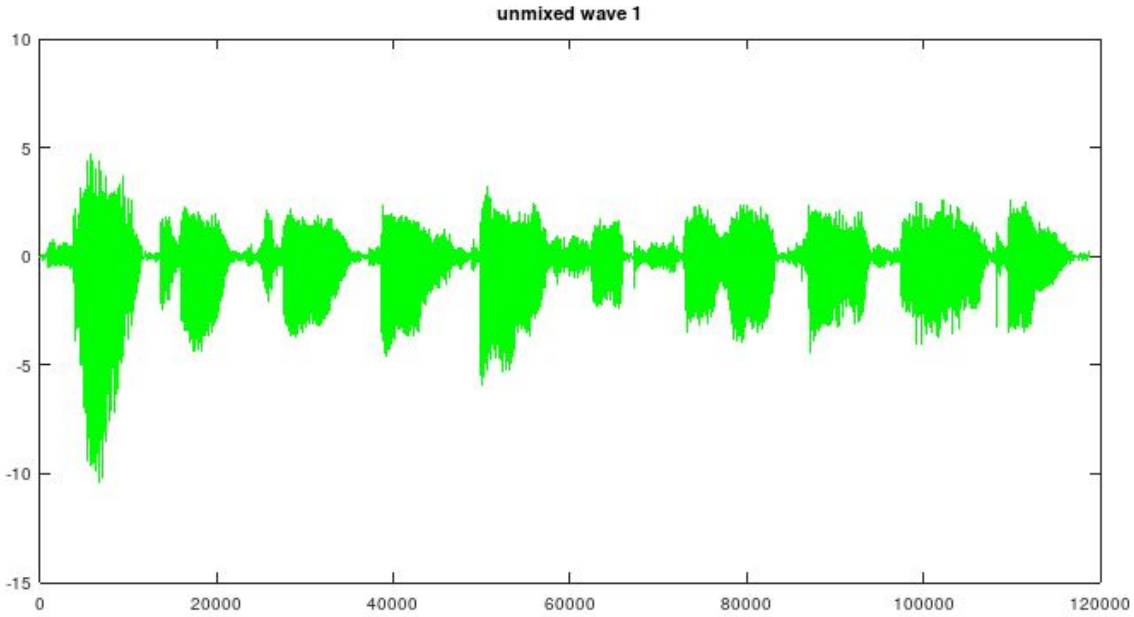


The audio recorded by microphone 1:[2]



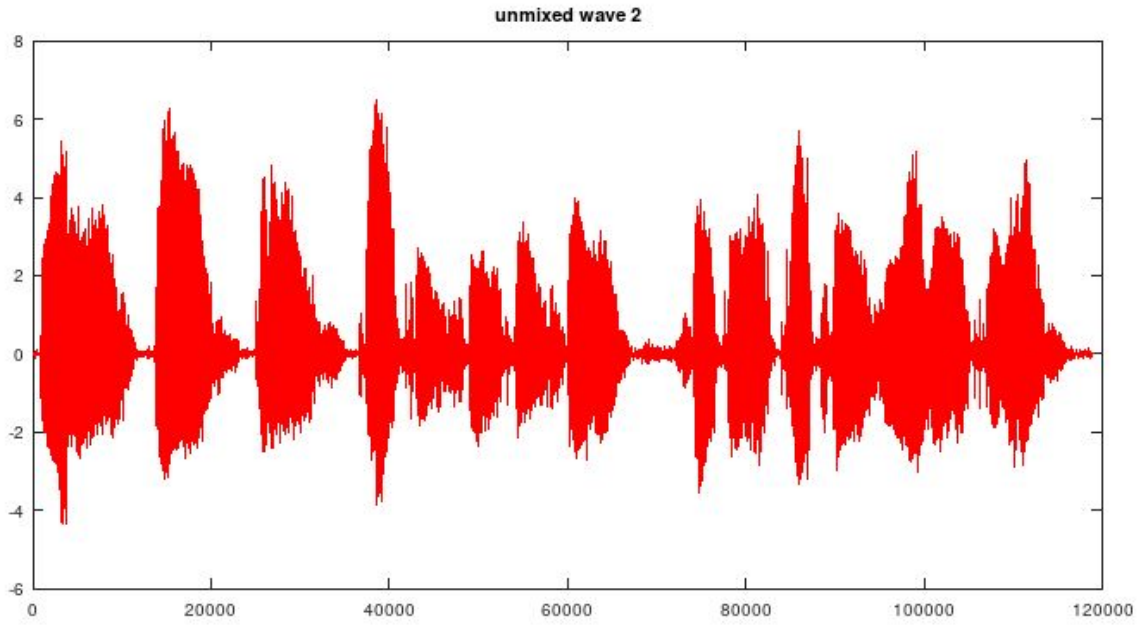
The above two audio graphs show the raw audio data[3] being recorded.

Separated Audio 1[2]:



The above graph is the separated audio of one user. It has a striking resemblance to both the raw audio files and further visual comparison to the raw audio graphs shows that the audio file received seems to be derived from the 2 graphs. Similarly, below is the graph of the audio file of the second user.

Separated Audio 2[2]



It is clear from both the graphs that they have been derived (as the matrix U in svd has been multiplied to the processed audio files) from the original microphone audio. The matrix multiplication, thus acts as a sieve through both the audio recordings filtering out one voice while enhancing the other.

The final scaling of both the audio tracks using the formula:

Output = Output * 0.1

Multiplying each element of the final output matrix by 0.1, then further enhances the separated (unmixed) matrix, by helping to remove the lower volume noise or unfiltered second voice in the audio.

Thus through the cocktail party algorithm, an audio input of N spatially separated microphones, recording audio from N spatially separated speakers leads to an audio output of N files with the voices of N separate speakers.

5. SUMMARY AND CONCLUSIONS

The end goal of the cocktail party algorithm and this paper is to try and replicate a simple and intuitive function of the human brain: To filter and focus on a single audio source in a crowd or a field of disturbances, such as is present in nature.

Over the six and a half decades since the problem has been introduced in Colin Cherry's classic 1953 paper[1] on the conundrum, there has been tremendous progress, but we are nowhere close to building a system as capable as a human or animal brain at isolating sounds in the environment.

In our approach above, we have a simple recording of 2 speakers on 2 different spatially separated microphones, but this system can be scaled to N speakers and N microphones, but we have observed a steady decline in quality of separation making large scale execution of such an algorithm unlikely.

The cocktail party problem's solution has many applications - from Natural language processing, helping isolate specific sounds in nature for wildlife research, smart speakers, smart AI assistants and so on. The benefits of isolating a particular sound from the rest have immense application.

6. BIBLIOGRAPHY

[1] Colin Cherry: Some Experiments on the Recognition of Speech, with One and with Two Ears
https://www.researchgate.net/publication/7752586_The_Cocktail_Party_Problem

[2] Cocktail party algorithm visualization
https://stackoverflow.com/questions/20414667/cocktail-party-algorithm-svd-implementation-in-one-line-of-code
https://stackoverflow.com/questions/51646592/cocktail-party-audio-source-separation

[3] Audio recordings from microphone
https://cnl.salk.edu/~tewon/Blind/blind_audio.html

[4] Study of the cocktail party algorithm: Abstract
https://ieeexplore.ieee.org/document/8251979

[5] Linear Algebra And Its Applications by Gilbert Strang (4th Edition)(Chapter 6) (Section 6.3)[Pages 367 - 373]

[6] Stanford cs229 course notes: Independent Component Analysis
http://cs229.stanford.edu/notes/cs229-notes11.pdf

[7] SVD equation in OCTAVE: Andrew Ng at Introduction to Machine Learning Course Coursera, Yair Weiss and Eero Simoncelli
https://cs.nyu.edu/~roweis/kica.html