Karim_Manisha__CAP_6673_004

In [1]:
```
import numpy as np
import pandas as pd
from scipy.io import loadmat
import matplotlib.pyplot as plt
from PIL import Image as im
from scipy.linalg import svd
```

In [18]:
```
img1 = im.open("Recording-1.png")
img1 = (np.asarray(img1)/255)

img2 = im.open("Recording-2.png")
img2 = (np.asarray(img1)/255)
```

In [3]:
```
img = [img1,img2]
```

In [4]:
```
img1
```

Out[4]:
```
array([[0.51372549, 0.4627451 , 0.89803922, ..., 0.45490196, 0.48235294,
        0.05490196]])
```

In [5]:
```
img2
```

Out[5]:
```
array([[0.00201461, 0.00181469, 0.00352172, ..., 0.00178393, 0.00189158,
        0.0002153 ]])
```

In [10]:
```
#PRE-PROCESSING

def center(x):
    mean = np.mean(x, axis=1, keepdims=True)
    centered =  x - mean
    return centered, mean

def covariance(x):
    mean = np.mean(x, axis=1, keepdims=True)
    n = np.shape(x)[1] - 1
    m = x - mean

    return (m.dot(m.T))/n

def whiten(X):
    # Calculate the covariance matrix
    coVarM = covariance(X)

    # Single value decoposition
    U, S, V = np.linalg.svd(coVarM)

    # Calculate diagonal matrix of eigenvalues
    d = np.diag(1.0 / np.sqrt(S))

    # Calculate whitening matrix
    whiteM = np.dot(U, np.dot(d, U.T))

    # Project onto whitening matrix
    Xw = np.dot(whiteM, X)

    return Xw, whiteM
```

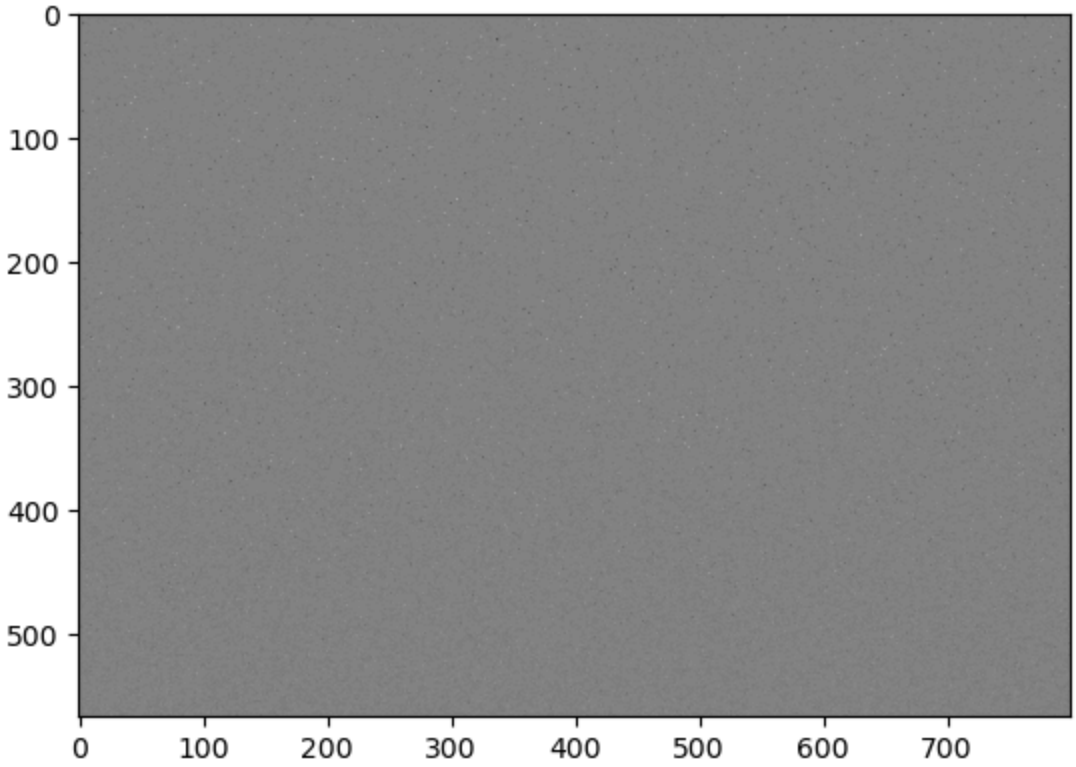
In [11]:
```
def Update_Weights(signals,  alpha = 1, thresh=1e-8, iterations=5000
0):
    m, n = signals.shape

    # Initialize random weights
    W = np.random.rand(m, m)

    for c in range(m):
            w = W[c, :].copy().reshape(m, 1)
            w = w / np.sqrt((w ** 2).sum())

            i = 0
            lim = 100
            while ((lim > thresh) & (i < iterations)):


                ws = np.dot(w.T, signals)
                wg = np.tanh(ws * alpha).T
                wg_ = (1 - np.square(np.tanh(ws))) * alpha

                # Update weights
                wNew = (signals * wg.T).mean(axis=1) - wg_.mean() *
w.squeeze()

                # Decorrelate weights
                wNew = wNew - np.dot(np.dot(wNew, W[:c].T), W[:c])
                wNew = wNew / np.sqrt((wNew ** 2).sum())

                # Calculate limit condition
                lim = np.abs(np.abs((wNew * w).sum()) - 1)

                # Update weights
                w = wNew

                # Update counter
                i += 1

            W[c, :] = w.T
    return W
```

In [13]:
```
def ICA(img):
    Xc, meanX = center(img)
    Xw, whiteM = whiten(Xc)
    W = Update_Weights(Xw,  alpha=1)

    recon = Xw.T.dot(W.T)
    recon = (recon.T - meanX).T

    return recon
```

In [19]:
```
recon1 = ICA(img1)
recon2 = ICA(img2)
```

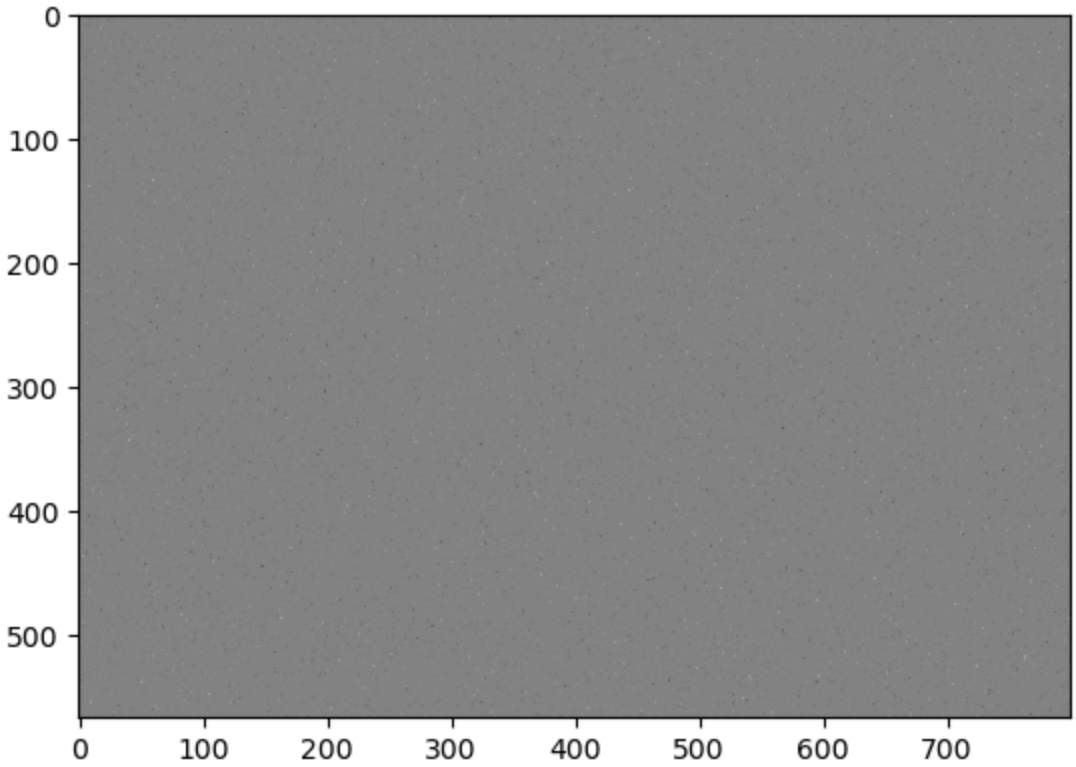
In [22]:
```
plt.imshow(recon1.reshape(567,800), cmap = 'gray')
```

Out[22]: <matplotlib.image.AxesImage at 0x7fcdd33a5490>



In [21]:
```
plt.imshow(recon2.reshape(567,800), cmap = 'gray')
```

Out[21]: <matplotlib.image.AxesImage at 0x7fcdd33ac2b0>



In [ ]:

In [ ]:

In [ ]:

In [ ]: