Synchronization

June 22, 2020

0.1 Using pilot signal

0.1.1 Steps:

1. Taking an image(MONALISA) and converting into bits.
2. Applying 4QAM Modulation scheme
3. Adding pilot signal of frequency 10khz and 50k samples
4. Converting it into .wav format for audio
5. Playing it through phone to computer's microphone.
6. Applying correlation to recieved signal and pilot signal
7. Demodulation
8. Final image and biterror rate

[ ]:
```
#imports
import numpy as np
from scipy.io.wavfile import write
import wave
from pylab import *
from scipy.io import wavfile
import matplotlib.pyplot as plt
from scipy import signal
```

[2]:
```
#Params
T=(10)**(-6)                  # Creates a row matrix with 11000 elements as␣
 ↪zeroes
T_s=2*(10**(-8))                  # Defining Time period of signal
f_c=2*(10**(6))                   # Defining Frequency of the signal
f_s=50*(10**(6))
```

[3]:
```
MonaLisa=np.load('binary_image.npy')
b = np.reshape(MonaLisa, (1,np.product(MonaLisa.shape)))[0] # b converts␣
 ↪(110,100) matrix to 1d array
T=(10)**(-6)
x=np.zeros(11000)                  # Creates a row matrix with 11000 elements␣
 ↪as zeroes
T_s=2*(10**(-8))                   # Defining Time period of signal
f_c=2*(10**(6))                    # Defining Frequency of the signal
f_s=50*(10**(6))                  # Defining Sampling frequency
```

```python
# This for loop encodes bits into constellations

for i in range (0,11000):
    if (b[i]==0):
        x[i]=1
    else:
        x[i]=(-1)



# Modulation
# Discrete time model
s_t=[]
for j in range (0,5500):           # Creates a waveform to transmit
    for n in range(50*(j),50*(j+1)):
        s_t.append(x[2*j]*(np.cos(2*(np.pi)*(f_c)*(n*(T_s))))+(x[(2*j)+1]*(np.
 ↪sin(2*(np.pi)*(f_c)*(n*(T_s))))))
```

[4]:
```python
# create new pilot signal of 10kHz FREQ
f_p = 10**4
T_s=2*(10**(-8))
pilot = []
for n in range(0,5*(10**4)):
    pilot.append(np.sin(2*np.pi*f_p*n*T_s))
```

[5]:
```python
transmit = np.concatenate([pilot,s_t])
```

[6]:
```python
print(len(transmit))
```

325000

[7]:
```python
transmit = np.array(transmit, dtype=np.float32)

samplerate = 44100
write("pilot_monalisa.wav", samplerate, transmit)
```

[13]:
```python
import pyaudio
import wave

# the file name output you want to record into
filename = "pilot_monalisa_rec_new1.wav"
# set the chunk size of 1024 samples
chunk = 1024
# sample format
FORMAT = pyaudio.paInt16
# mono, change to 2 if you want stereo
```

```
channels = 1
# 44100 samples per second
sample_rate = 44100
record_seconds = 9
# initialize PyAudio object
p = pyaudio.PyAudio()
# open stream object as input & output
stream = p.open(format=FORMAT,
                channels=channels,
                rate=sample_rate,
                input=True,
                output=True,
                frames_per_buffer=chunk)
frames = []
print("Recording...")
for i in range(int(44100 / chunk * record_seconds)):
    data = stream.read(chunk)
    # if you want to hear your voice while recording
    # stream.write(data)
    frames.append(data)
print("Finished recording.")
# stop and close stream
stream.stop_stream()
stream.close()
# terminate pyaudio object
p.terminate()
# save audio file
# open the file in 'write bytes' mode
wf = wave.open(filename, "wb")
# set the channels
wf.setnchannels(channels)
# set the sample format
wf.setsampwidth(p.get_sample_size(FORMAT))
# set the sample rate
wf.setframerate(sample_rate)
# write the frames as bytes
wf.writeframes(b"".join(frames))
# close the file
wf.close()
```

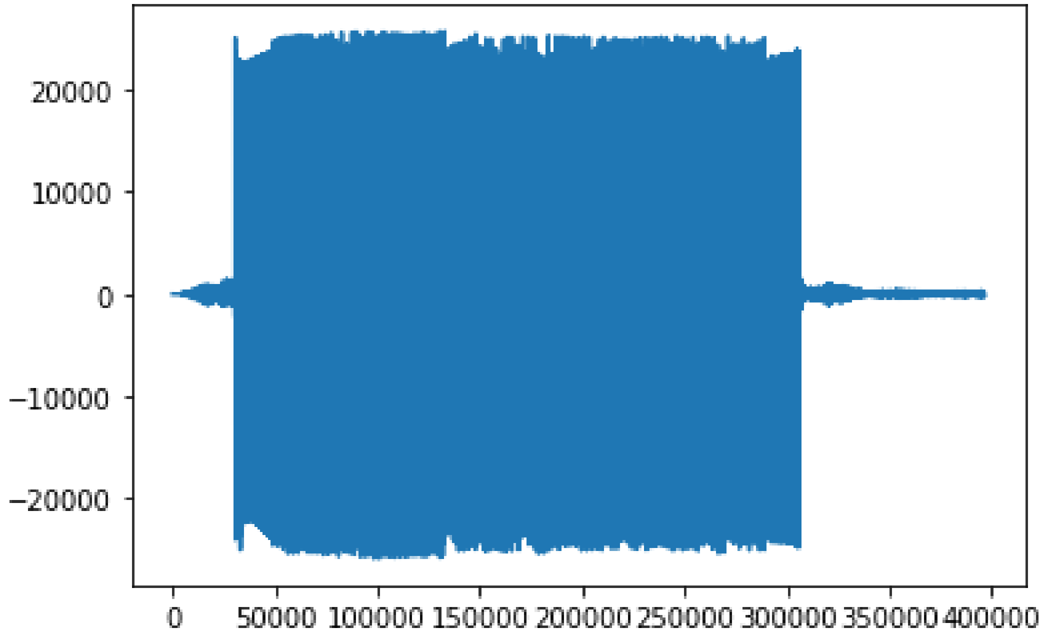
```
Recording...
Finished recording.
```

[14]:
```
fs, rec_pilot = wavfile.read('pilot_monalisa_rec_new1.wav')
```

[15]:
```
print(len(rec_pilot))
plt.plot(rec_pilot)
```

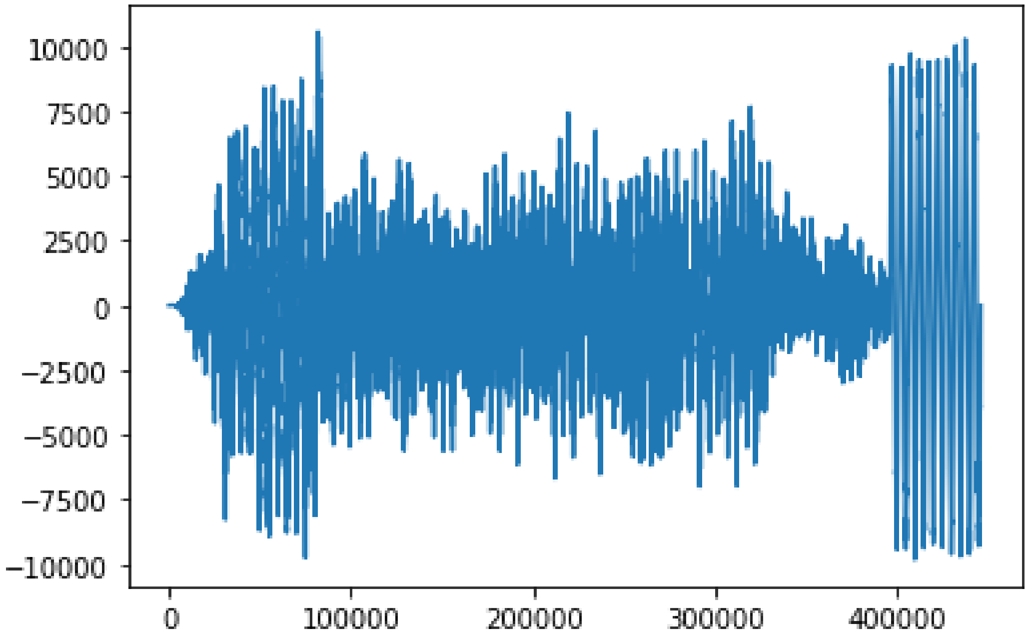
```
plt.show()
```

396288



[16]:
```
correlation = signal.correlate(rec_pilot,pilot)
```

[17]:
```
plt.plot(correlation)
plt.show()
```



[18]:
```
maxim = np.amax(correlation)
print(maxim)
result_new = np.where(correlation == np.amax(correlation))
print(result_new)
```

```
10560.094312901472
(array([81963], dtype=int64),)
```

[36]:
```
final_recieved = rec_pilot[81963:356963]
print(len(final_recieved))
r = final_recieved
```

```
275000
```

[37]:
```
# Demodulation by minimum distance decoding
# Creating four signals s1,s2,s3,s4 as constellation in 4-QAM modulation scheme
s1=np.zeros(275000)
s2=np.zeros(275000)
s3=np.zeros(275000)
s4=np.zeros(275000)
# u1,u2,u3,u4 are distances of r from constellations s1,s2,s3,s4 respectively
u1=np.zeros(5500)
u2=np.zeros(5500)
u3=np.zeros(5500)
u4=np.zeros(5500)
# This for loop calculate distances u1,u2,u3,u4
```

```
# Calcualting distances of 50 samples from both s_i(i=1,2,3,4) and r and storing␣
↪it in u1,u2,u3,u4 respectively

for e in range(0,5500):
    for n in range(50*(e),50*(e+1)):
        s1[n]=((np.cos(2*(np.pi)*(f_c)*(n*(T_s))))+(np.sin(2*(np.
↪pi)*(f_c)*(n*(T_s)))))
        s2[n]=(np.cos(2*(np.pi)*(f_c)*(n*(T_s))))-(np.sin(2*(np.
↪pi)*(f_c)*(n*(T_s))))
        s3[n]=-(np.cos(2*(np.pi)*(f_c)*(n*(T_s))))+(np.sin(2*(np.
↪pi)*(f_c)*(n*(T_s))))
        s4[n]=-(np.cos(2*(np.pi)*(f_c)*(n*(T_s))))-(np.sin(2*(np.
↪pi)*(f_c)*(n*(T_s))))
        u1[e]=u1[e]+(r[n]-s1[n])**2
        u2[e]=u2[e]+(r[n]-s2[n])**2
        u3[e]=u3[e]+(r[n]-s3[n])**2
        u4[e]=u4[e]+(r[n]-s4[n])**2

# Taking the minimum values from [u1[i],u2[i],u3[i],u4[i] ] for i from 0 to 5499
y_1=[]
for o in range(0,5500):
    y=[u1[o],u2[o],u3[o],u4[o]]
    y_1.append(min(y))


# Codes u1,u2,u3,u4 to 1,2,3,4 and stores the values in y_2
y_2=[]
for h in range(0,5500):
    if (y_1[h]== u1[h]):
        y_2.append(1)
    elif (y_1[h]==u2[h]):
        y_2.append(2)
    elif (y_1[h]==u3[h]):
        y_2.append(3)
    elif (y_1[h]==u4[h]):
        y_2.append(4)
c=np.zeros(11000)

# assigning bits to corresponding constellation points and stores it in array c
for p in range(0,5500):
    if (y_2[p]==1):
        c[2*p]=0
        c[2*(p)+1]=0
    elif (y_2[p]==2):
        c[2*p]=0
        c[2*(p)+1]=1
    elif (y_2[p]==3):
```

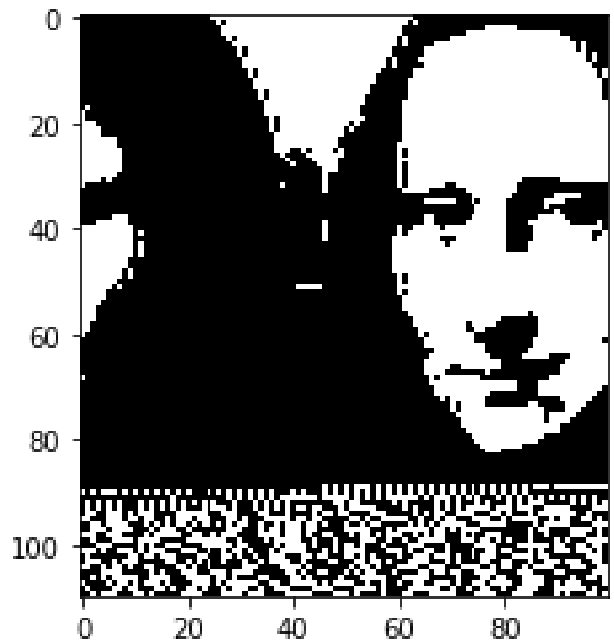
```
        c[2*p]=1
        c[2*(p)+1]=0
    elif (y_2[p]==4):
        c[2*p]=1
        c[2*(p)+1]=1

# Reshapes the array c to matrix(110,100)
d = c.reshape(110,100)


#Calculates bit error rate and no.of error bits
z_1=np.zeros(11000)
for q in range(0,11000):
    if (c[q]==b[q]):
        z_1[q]=0
    else :
        z_1[q]=1
k_1=np.count_nonzero(z_1)
k=0
for i in range(0,11000):
    k=k+z_1[i]
print('The number of error bits is:',k_1)
print('The bit error rate is:',k/11000)

# plots the final image
plt.imshow(d,'gray')
plt.show()
```

```
The number of error bits is: 6167
The bit error rate is: 0.5606363636363636
```


0
20
40
60
80
100
0
20
40
60
80

[ ]: