Utah State University
ECE 5660
Communication Systems I
Final Programming Project
Due Monday April 24, 2024 (last day of class)

The program `com_sim_for_students_2024.py` illustrates reading in an image file and modulating it. Each pixel is represented using one byte of data (8 bits) to form a grayscale image. For comparison, the program `com_sim_for_students_2024.m` performs the same function using Matlab.

The file `kitten1_8bit.dat` is provided. This is a binary file representing an image with the following format:

```
rows cols bits(1) bits(2) ... bits(r*c)
```

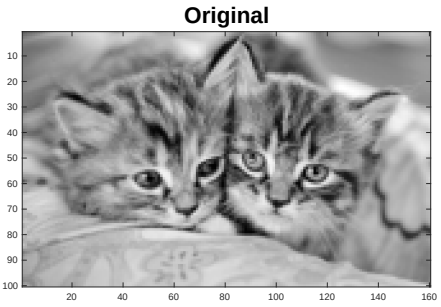
where

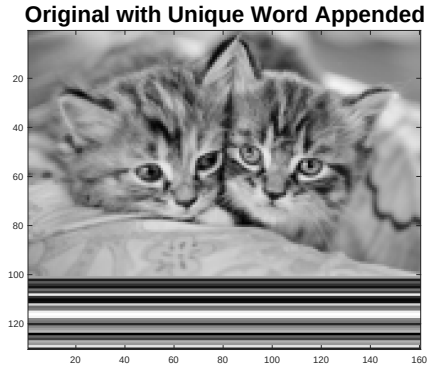
`rows` is the number of rows in the image (32-bit signed integer)

`cols` is the number of columns in the image (32-bit signed integer)

`bits(k)` are the 8-bit patterns that represent the pixel intensities of the image. These 8-bit patterns are used to select the symbol to transmit using a 256×2 lookup table for a QAM signal constellation.

This file is read into the program. As displayed (using `imshow` in Python or `imagesc` in Matlab, the image looks like the image below on the left:





original image image with UW

After the image is read in, a unique word (UW) is added to each column. The image with these unique words is shown above on the right.

The image with the UW is scanned by columns to produce a sequence of bytes. These are modulated using 256-QAM. Since 256-QAM is used, each symbol carries 8 bits of data, obviating the need to split the bytes into smaller units for modulation.

The modulated data is saved in the file `test_2024` as an ascii (that is, you could look at this using a text editor.) file, series of real numbers representing the modulated signal. As saved in the file, there is no offset, no timing offset, no frequency offset, and no clock timing offset: this is the simplest of possible circumstances to test on. (You can adjust parameters in `com_sim_for_students_2024.py` to generate other examples.) This data is provided for you to test your programs.

In addition, there are other data files on the canvas website. These are all ascii files. These are modulated with the signal distortions as described here:

(practice) `test_2024` – no timing or phase offsets. Produced by `com_sim_for_students`.

(10 pts) `sim1_2024` – no timing or phase offsets

(15 pts) `sim2_2024` – fixed phase offset, no timing offset

(20 pts) `sim3_2024` – carrier frequency offset, no timing offset

(15 pts) `sim4_2024` – no phase offset, fixed timing offset

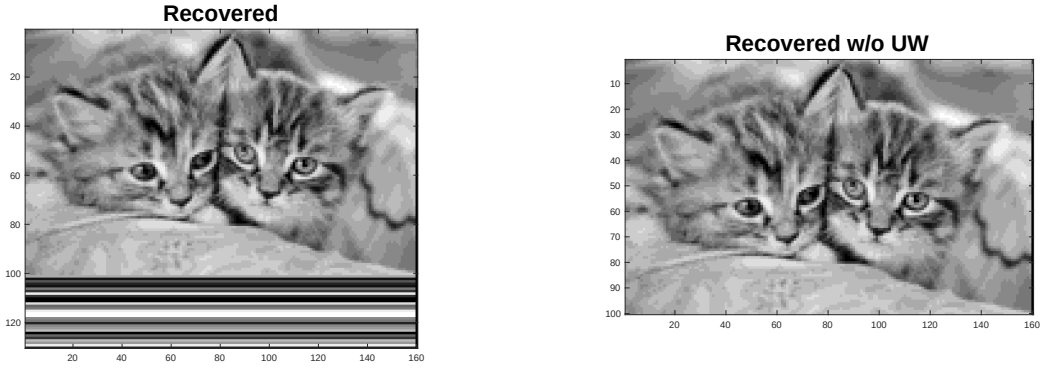
(20 pts) `sim5_2024` – no phase offset, timing clock frequency offset

(40 pts) `sim6_2024` – carrier frequency and timing clock frequency offsets

Full points will be provided if you turn in the plots for each case, as described below.

Your assignment is to write a PYTHON program that performs joint carrier phase and symbol timing recovery, and then to apply that program to the each of the files given above.

As an example of the processing, the image on the left below shows the recovered image. The recovery is such that the UW is still at the bottom of the image. In the image on the right the UW has been removed.



recovered image with UW output image with UW removed

It is not necessary to remove the UW, but you may want to do this. Your goal is to produce a decoded image for all six of the images above as well as possible.

In your code, you will get an extra 5% per image if you arrange the image so that the UW appears at the bottom of the image (as in the example). This will require some extra thought.

To obtain full points for the items above, you must include the items identified below, as appropriate for the part.

1. For each simulation in which phase recovery is required (2,3,6), plot the phase estimate $\hat{\theta}$ and the filtered phase error signal (phase loop filter output).
2. For each simulation in which timing recovery is required (4,5,6), plot the filtered timing error signal (timing loop filter output) and μ (the fractional interval). Use dots for these plots.
3. For each simulation, plot the decision variables (that is, the phase-aligned, interpolated values) in the signal space as a scatter plot (use dots).
4. For each simulation, provide a picture of the image you recovered.

Make sure your project includes the following:

- (5 pts) Draw a detailed block diagram of your receiver system, including both phase and timing synchronization.
- (10 pts) Turn in listings of your *Python* code.
- (10 pts) Provide a discussion of what you learned, what difficulties you had, and how you were able to debug your system.
- (5 pts) Make a table listing the parameters and values of K1, K2, BnTs, zeta, N, K0, Kp (and other parameters you may have selected). Explain how you chose the values for these parameters.