Onyedikachi Okemiri

How To Edit the Radar Code For You

Most of the heavy lifting, so to speak, has been completed in terms of the overall signal processing. The code provided has 2 version. One of them uses a stereo audio file as the input and the other one uses a 2 column CSV file. Other than the file inputs those two versions are the same. This guide should help you navigate the code more easily. This will be for the matlab version of the code, but very similar to the python version.

So to start off you want to make sure you are inputting the right file.

```
[Y,FS,NBITS] = wavread('running_outside_20ms.wav');
```

This line will allow you to change which file you wish to read in. Changing the name inside of the single quotation marks will change the data you are reading. Be sure to match the name for each file and try to keep the file names meaningful and unique. It helps when you are testing multiple scenarios so that you can find your data more easily. One extra thing to note is that there should be two channels when inputting your data. The first channel should be your sync signal and the second channel should be the data received by your radar system. Double check to make sure both signals are there because they are both required for the code to work properly.

Next up there are a few variables that will be recurring throughout the code. The only variables you will need to change are Tp, fstart and fstop. These values will be unique to your own system. In order to obtain fstop and fstart you need to find the range your VCO is outputting based on the voltage of the triangle wave input. You will have to refer to the VCO datasheet to get an accurate range for the frequencies. Tp will be determined by the frequency of your VCO input signal. Tp = (frequency/2)^(-1). This is extremely important for obtaining correct results from your plots.

The middle part of the code doesn't need much attention, but in quick summary, it takes the rising edges of the sync pulses to parse the data. It takes the entire string of data and turns them into smaller samples to analyze. It then takes the average of all the data and creates a new array based on the highest readings.

Moving on to the end of the code we now see where the plots are generated. There is a bit more manipulation you can do to try and get cleaner plots. By default the code is set up to give you two plots: `RTI without clutter rejection and RTI with 2-pulse cancelor clutter rejection.` From my experience the plot without clutter rejection is better for stationary measurements while latter is better for moving measurements.

```
imagesc(linspace(0,max_range,zpad),time,S-m,[-80, 0]);
```

If you are interested in obtaining more accurate results from the stationary plot change **S-m** to **mean(S-m).** This helps to more easily see where peaks in signal are along with their distances. Often times in the early stages of radar testing you will have to be indoors when testing. This will prove to be very difficult because it will introduce many reflections and overall noise to the system and the end plot. One way to clean up the plots is to shift the scale of plots. From the start the plots are set to show a scale from 0dB to -80dB. However, when in the lab many of the

reflections will be very high in signal power so you can change the scale to something like 0dB to -40dB. This will scale the lowest signal you see to -40dB and make it a drastically different color from the higher power signal even though they may be fairly close in magnitude. This is also true in reverse. When you are able to test your system outside at much further distances, you can employ a similar tactic. Your signals will naturally be much lower so dropping the highest signal to something like -10 or -20dB will allow you to scale weaker signals up and give them a different color. This makes viewing measurements at a distance much easier and cleaner. The values I provided are rough estimates, so you will have to tweak these according to what works best for you.

That about wraps up modifying the code. As stated, there are things you are able to do to get the best out of this code, but it just takes some creativity to do them. If you are familiar with matlab or python there may be even more techniques to clean up the results to give you the most accurate data.