

PRACTICAL SIGNAL DETECTION AND CLASSIFICATION IN GNU RADIO

Timothy J. O'Shea (NC State University, Raleigh, NC; tim.oshea@ieee.org);
T. Charles Clancy (Department of Defense, College Park, MD; clancy@LTSnet.net);
Hani J. Ebeid (University of Texas, Austin, TX; HJEbeid@utexas.edu)

ABSTRACT

This paper selects a number of effective, general methods for enabling signal detection, estimation, and classification needs for cognitive radio. Implementations of algorithms such as these are of key importance to Dynamic Spectrum Access (DSA). These algorithms are decomposed into logical blocks and then implemented in reusable GNU Radio signal processing blocks. These blocks are then demonstrated in an example GNU Radio application running in a Linux environment, using a Universal Software Radio Peripheral (USRP) as a radio frontend. A second USRP on an unconnected host computer is used to generate the relevant test signals used for training and detection/classification trials.

A key advancement presented in this work is application of these algorithms to real-world signals input from an RF frontend, rather than ideal signals generated in MATLAB. We show that work is needed to normalize the output of the USRP to make signal detection and classification more robust.

1. INTRODUCTION

One of the most popular applications of cognitive radio is that of Dynamic Spectrum Access (DSA). In DSA, radios must monitor activity on a given segment of radio-frequency (RF) spectrum and attempt to identify available unused regions: regions belonging to primary user signals whose service levels must not be degraded, and regions used by other secondary users' signals with which we may wish to communicate in order to form a cognitive radio network.

For the purposes of distinguishing between primary and secondary user signals, as well as establishing meaningful communications with other secondary users, and effective method for classification of observed signal modulation is needed. This paper will focus on selecting effective algorithms which have been presented in prior research, implementing and connecting them to lay the detection and classification foundation for a DSA capable cognitive radio built by expanding upon the tools included in the GNU Radio Project. The Universal Software Radio Peripheral (USRP) and the Cell Microprocessor are targeted as an ideal

combination platform for this architecture due to their capabilities, low-cost, and wide-spread availability.

The remainder of the paper is organized as follows. Section two discusses the system architecture within GNU Radio. Section three details our experimental, laboratory results. Section four outlines avenues of future research and improvements to our implementation. Section five concludes.

2. SYSTEM ARCHITECTURE

We can generally divide the task of receiving and characterizing the observed RF into three different logical groupings. The first consists of a number of generic receiver functions which condition the received signal for further processing or demodulation. The second consists of signal detection and bandwidth estimation schemes. The third consists of our signal modulation classification task.

The implementation and interaction between these components will depend on the specific GNU Radio structure. The ultimate method in which GNU Radio will be structured to take advantage of highly parallel platforms such as the Cell Microprocessor has not yet been determined. Consequently, we offer design considerations which should allow our implementation to remain effective should any of these methods be chosen.

2.1. GNU Radio on the Cell Microprocessor

GNU Radio consists of a number of radio processing components referred to as blocks, which may be linked together to form a useful waveform. Traditionally, each of these blocks run in a single thread and a scheduler has been used to run each block's work task when it has a non-empty input queue. However in order to take advantage of the Cell architecture, we will need to distribute this work load onto multiple processors. This can be done through the expected long-term approach of loading each block onto a Synergistic Processing Elements (SPE) to form a traditional pipeline, or through the short-term approach of simply offloading the work from one or two distinct processor intensive blocks onto available SPEs.

In the short term approach, as shown in Figure 1, all blocks are actually executed on the Power Processing

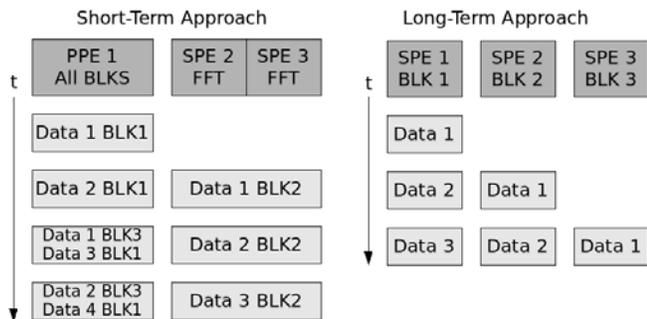


Figure 1: Proposed GNU Radio on Cell Implementation.

Element (PPE). In the diagram, we show the movement of both data, denoted Data x , through processing code blocks, denoted BLK y . The GNU Radio blocks are threaded on the PPE, and when BLK 1 finishes processing Data 1, BLK 2 starts. It makes a blocking call to task one or more SPEs with performing some computation, such as an FFT. While this thread is blocking, the BLK 1 thread is free to begin preparing Data 2 until the SPEs return.

The short-term method is primarily the approach we will be targeting, as we attempt to parallelize as much of the workload into PPE threads as possible and make blocking SPE-based calls to do the heavy work.

The long-term approach also pictured in Figure 1 should distribute an equal work load to each SPE, perhaps by grouping logical signal processing blocks, and ideally using the Cell's ring-bus topology in a linear fashion. However this approach poses many additional challenges with dividing and scheduling workloads which must be addressed first; therefore, we will focus primarily on the first method when considering our design.

2.2. Common Front End

The USRP combined with an appropriate daughterboard (we used an 800-2400 Mhz board) provides the ability to downconvert our tunable frequency band to IF, digitize the signal at 64 MSPS using an AD9862 Mixed-Signal Front-End Processor, and decimate this down to an appropriate rate that we can move it over the 480Mbps USB 2.0 interface.

The USRP is fully supported by GNU Radio, and very little effort is required to add the appropriate source block into our waveform. Since we can receive the decimated IF for an arbitrary tunable center frequency decimated at a selectable rate using this USRP source block, all that remains is to ensure that we maintain an appropriate dynamic range by adjusting the programmable gain amplifier (PGA) on the AD9862. PGA control is implemented as shown in Figure 2. It uses the GNU Radio standard Stream-to-Streams, Serial-to-Parallel, and Keep-

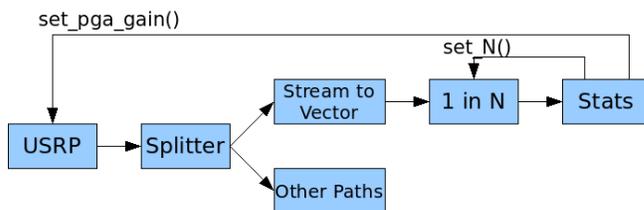


Figure 2: Common Signal Conditioning Pathway

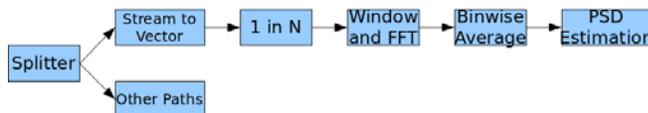


Figure 3: Signal Detection Pathway

One-In- N blocks. Additionally, a free-running python thread generates statistics on the samples, and steps the PGA up or down based on the sample vector mean falling above or below preset thresholds. Lastly, the N parameter of the vector decimation block, or period between updates measured in vectors of samples, is modified when we observe the correct or incorrect dynamic signal range. This quickly achieves an ideal gain value while relinquishing processing resources when not needed.

	<u>Start Frequency</u>	<u>Stop Frequency</u>
Hole 1	2396031250	2399875000
Signal 1	2399937500	2400062500
Hole 2	2400171875	2403984375

Table 1: Values Inserted into RF Map

2.3. Signal Detection Block

The signal detection block operates as a simple energy detector, performing thresholding and estimation on the output of a time-averaged power spectral density (PSD). The layout of this pathway is shown in Figure 3.

The signal detection pathway begins by vectorizing samples into groups of 512, and decimating the vectors to a rate which we can sustain real-time processing. We then apply a Blackman-Harris Window to each 512-sample vector and take a 1-D, complex FFT, averaging the magnitudes of each bin over many samples in the next block. Finally, in the last block we calculate the mean and variance of the averaged PSD in the estimation step, artificially increasing the variance up to some minimum level, handling the case of no signal being present. We then establish two thresholds using these statistics, and divide the frequency

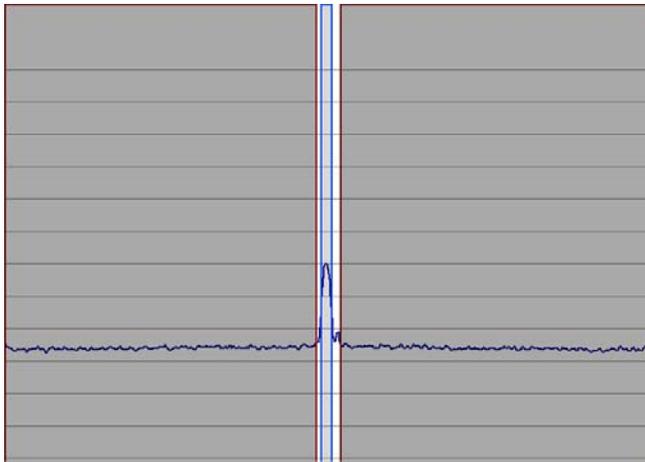


Figure 4: Energy Detection on Observed Spectral Region

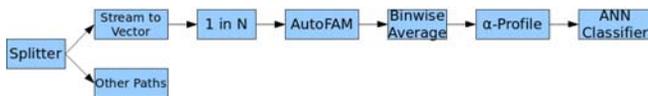


Figure 5: Signal Modulation Classification Pathway

axis up into regions based on whether we fall under or over these thresholds. The regions are defined by the following:

$$\begin{aligned} \text{Confident free spectrum:} & \quad P(f) < \mu + 0.2\sigma \\ \text{Confident signal spectrum:} & \quad P(f) > \mu + 3\sigma \end{aligned}$$

By using these regions to classify any given frequency bin, and forming regions out of consecutive bins, we are able to quickly characterize the observed signal space. In the example shown in Figure 4 we see a GMSK signal centered at 2.4 GHz being generated by another USRP across the room. The observed regions are highlighted, and the corresponding additions to the RF map are shown in Table 1.

Ultimately, a form of successive approximation should be used during this step to subtract recognized signals from the PSD, recalculate the statistics, and search for more regions, until we hit our artificially increased sigma value. The RF Map component which maintains a listing of the various signal and hole regions should perform logical unions and collision checking on signal and hole regions. However for our purposes, a single pass is used currently used during this step and provides a sufficient metric to detect most signals without issue. From the region start and end bounds we estimate the signal center frequency and bandwidth by calculating the center as the mean of f_{end} and f_{start} and the width as $f_{end} - f_{start}$.

This is a rough process, which could certainly be improved by a successive approximation algorithm which re-tuned around the suspected center, increased the

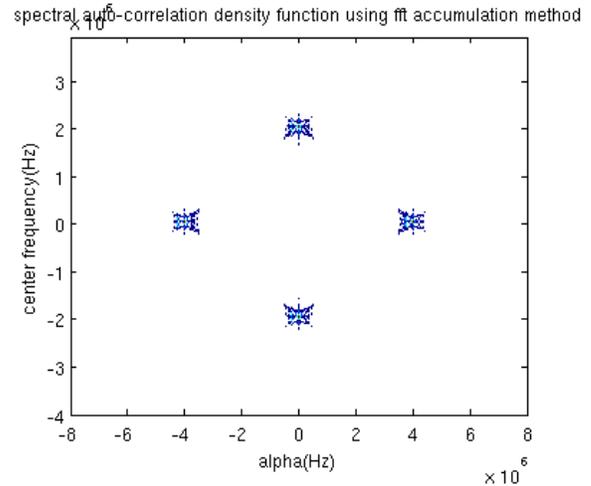


Figure 6: SCD for QPSK

decimation rate, and repeated the process until it achieved the desired resolution. For our purposes a single pass was implemented and robust estimation of fine signal movement or bandwidth adjustment was not heavily tested.

2.4. Signal Classification Block

After insertion of a signal region into the RF Map as an unclassified region, the classification control thread is signaled. This section immediately re-centers the RF tuner on the signal and sets the appropriate decimation on the front end to maximize the achieved resolution of our signal in the observation window. The energy detection flow graph is then paused while the classification pathway runs on the signal. This pathway is shown in Figure 5, and consists of the common task of breaking IF samples into vectors and reducing the data rate to something manageable, followed by a component which looks for cyclostationary features in the input signal by using the FFT Accumulation Method to calculate the Spectral Correlation Density (SCD) Function of the observed signal. This is then reduced to the alpha-profile, and sent into the ANN classifier block for a decision.

As has been demonstrated [1], each known modulation type will produce a different arrangement of peaks in its SCD plot, due to both inter- and intra-symbol correlation within the waveform. We can see in Figure 6 the ideal SCD plot of a QPSK signal averaged over many sampling periods. This arrangement of four clusters is common in many modulations, and this scale does not provide enough detail to highlight differences in the plots for various modulations.

Figures 7 and 8 compare an enlarged view of the rightmost major detail on the alpha axis of the SCD plot for both a QPSK signal and 4-FSK signal. The cyclostationary

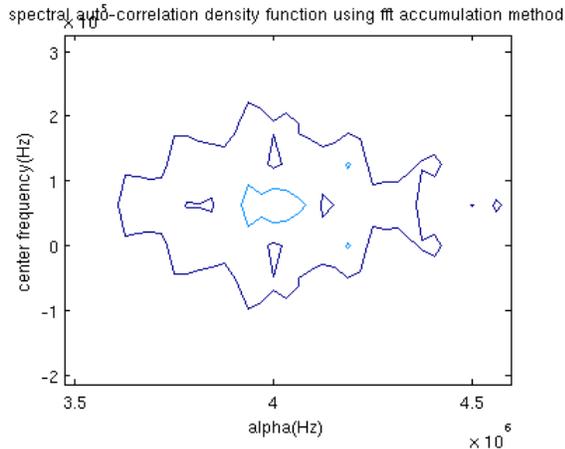


Figure 7: Detailed SCD for a 4-FSK Signal

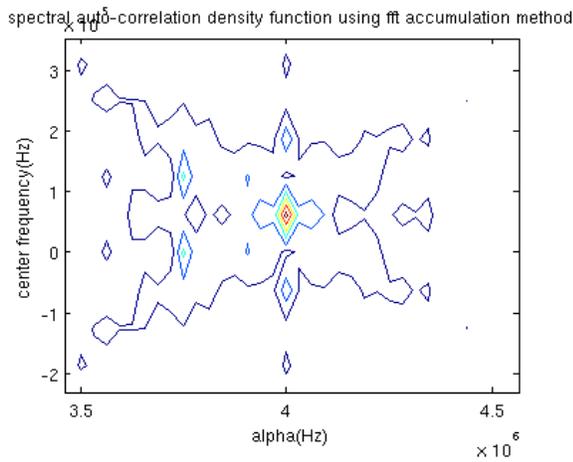


Figure 8: Detailed SCD for a QPSK Signal

properties of each of these signals are clearly different by observation. In order to allow our classifier to learn various modulation methods and discern between them, we will use the method proposed by Fehske, Gaeddert, and Reed [2] and take the alpha profile of our SCD plot as a 1-D vector input to an Artificial Neural Network (ANN) using the structure of a Feed Forward Multi-Layer Perception Network (MLPN) with back propagation [4].

Our input layer consists of 129 neurons, each one associated with a normalized bin value from the alpha profile of our SCD. As in [2]'s design, we use a hidden layer consisting of four hidden neurons, a learning rate of 0.05, a learning momentum of 0.7, and a sigmoid symmetric activation function for all neurons. Since we are initially trying to classify only the three digital modulation types currently supported for transmission under GNU Radio (DBPSK, DQPSK, and GMSK), we will use a three neuron output layer, with an orthogonal output vector associated

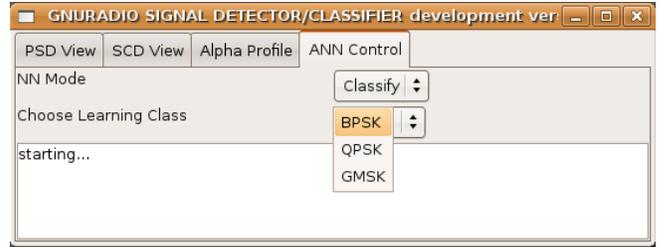


Figure 9: Artificial Neural Network Training Interface

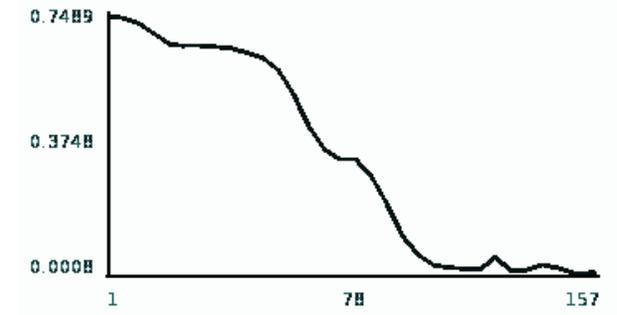


Figure 10: Mean Square Error Durring MLPN Training

with each of the three modulation types. Training data for the ANN is generated at run time through a graphical interface which allows the user to specify which type of modulation. It places the ANN in a learning mode, in which it outputs the appropriate input and output layer data. Figure 9 shows a screen capture of the ANN training interface.

Training the ANN weights at runtime proved to quickly skew the data towards the class with the most trials. To fix this we ultimately opted to write the training trials to a file where they could be manually inspected and adjusted if necessary for fairness. Upon initialization of the Classifier block in GNU Radio, these trials are then read in from the file and weights are calculated. The MSE of our output vector, as we progress through this training progress, is shown in Figure 10. This block uses the Fast Artificial Neural Network (FANN) library to implement, train, and execute the MLPN. FANN was chosen as it is already one of the fastest implementations available for this purpose, and the Vector Fast Artificial Neural Network (VFANN) project plans to accelerate this even more in the near future by using vector operations available on the graphics processing units (GPU's) in inexpensive, widely-available video cards, which we hope will be easily portable to the fast vector floating point operations available on the Cell Processor's SPEs. The output vectors used for training consist of permutations of $\{-1, -1, +1\}$, so to select the modulation chosen by this classifier we must simply choose the index of $\max(output_vector)$.

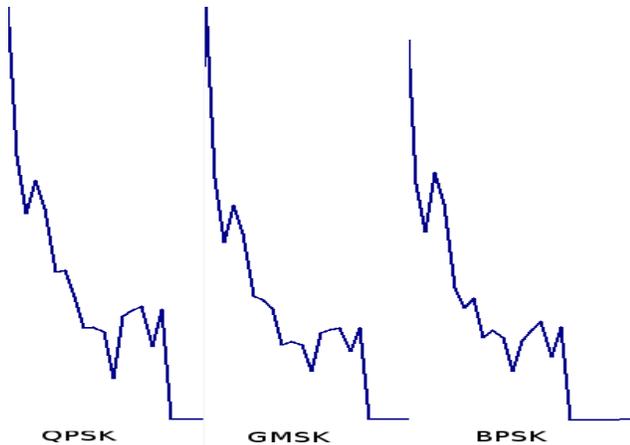


Figure 11: Comparison of Alpha-Profiles of Three Modulations

3. RESULTS

This design worked extremely well classifying signals, with the exception of a few minor issues. It did not make sense to present our results in the traditional SNR vs. detection rate fashion because most of the significant observable error could be attributed to several known factors which greatly outweighed the false detection rate inherent to the actual analytical method used. When these effects were not observed, we received a correct signal classification an overwhelming majority of the time (>95%). For more information on the theoretical limits of using this classification technique in varying SNR environments please refer to [2] which explores this topic in much more detail.

The PSD of the three narrow-band GNU Radio digital modulations (DBPSK, DQPSK, and GMSK) were trained on the order of minutes using a random stream of symbols as input. They were all trained at a single signal level of another stationary USRP transmitting from across the room at the GNU Radio tx-amplitude $3e5$. Therefore, this training occurred in a typical indoor noise environment with a roughly constant SNR of 30 dB.

When manually adjusting the transmit signal power on the transmitting USRP, we were able to retain our classification ability for a wide range of observed SNR values using our initial training. However, when attempting to move the signal to another frequency or bandwidth, we ran into issues. Since the classification block relies on the center frequency and bandwidth of the signal to be normalized when it receives its conditioned input, the resolution to which we are tuning and decimating is not fine enough, and we can observe gaps where fine changes in this value result in movement of the signal within the classification observation space. When making fine adjustments to the bandwidth or frequency from those at which it was trained, we can observe variable levels of

misclassification. However, typically if we fall within the center of an FFT bin, and with a diatomic multiple of the trained signal bandwidth (which does not fall outside the minimum and maximum decimation constraints imposed by the USRP) we are able to successfully classify. Additionally, small variations in the automatic gain control which were not seen during training often lead to misclassification of signals.

The resolution used in calculating our SCD was limited by GNU Radio buffer constraints, and our AutoFAM output was limited to 17×129 , which was sufficient for the successful classification described above. However, the majority of the information used in making the classification decision is localized in a few common areas of the plot, where higher resolution would most likely contribute to our correct classification rate. Figure 11 shows the similarity of a small interest region of the alpha-profile for the three modulation types.

4. FUTURE WORK

If this approach of performing classification on a completely normalized signal is to be effective and robust in the long term, several enhancements to this design are needed. Due to the coarseness of the adjustments provided by the programmable gain amplifier and diatomic, integer-only decimation values allowed by the USRP front-end, either much more fine control of these is needed at the front end. Alternatively, another layer of software re-sampling and signal-level normalization is required to condition the signal for the classification pathway.

An alternative to this may be to train the ANN with the RF frontend in a variety of possible configurations. Much as was done in [2] to train against signals at varying power levels, we could train over the expected fine range of bandwidths one would see between two coarse decimation values. This could be done for both signal amplitudes between two coarse programmable gain amplifier levels and for signal center frequencies between the center frequencies of two separate frequency bins. In these cases, as well as the case of an increased output vector size due to more modulation classes, we will need to re-evaluate the structure of the MLPN to allow for more degrees of freedom.

Another possible area to look at is using the output of the MLPN to estimate a confidence estimate, which could be useful in determining if we are seeing one of the modulation classes we have trained against, or possibly something which we have never seen before.

Additionally, the issue of low resolution SCD output could be addressed in a number of ways. One technique is to serialize transfer of data between blocks, or combining the data reduction of the alpha profile with the AutoFAM component. Additional methods for increasing resolution in various areas of the SCD plot could increase our resolution

and useful information upon which to base our classification.

5. CONCLUSION

In this paper we have taken a variety of signal detection and classification algorithms and implemented them within the GNU Radio architecture. We looked at real-world implementation issues that arise from operating with a real RF receiver/digitizer system, such as the USRP.

Many of the simulations completed as a part of current implementations do not have to deal with the peculiarities of automatic gain controllers, coarse signal decimation, and center-frequency tuning. We showed that a real-world implementation of these signal detection and classification algorithms, such as would be necessary for dynamic spectrum access radios, requires classification algorithms to either be robust to these variations, or first normalize signals prior to classification. Next steps for our research include implementing this required normalization.

Overall, we have shown that generic signal detection and classification is achievable in GPP-based SDR systems, but may the processing power associated with higher-performance GPPs, such as the IBM Cell Processor.

REFERENCES

- [1] E. L. Da Costa, "Detection and Identification of Cyclostationary Signals," *IEEE Transaction on Signal Processing*, 1996.
- [2] A. Fehske, J. Gaeddert, and J. H. Reed, "A New Approach to Signal Classification Using Spectral Correlation and Neural Networks," *Proceedings IEEE DySPAN*, pp 144-150, November 2005.
- [3] T. Clancy, J. Hecker, E. Stuntebeck, T. O'Shea. "Applications of Machine Learning to Cognitive Radio Networks," *IEEE Wireless Communications Magazine*, August 2007.
- [4] J. A. Anderson, *An Introduction to Neural Networks*, MIT Press, Cambridge, MA, 1995.