FROM SIMULINK TO SMARTPHONE: SIGNAL PROCESSING APPLICATION EXAMPLES

R. Pourreza-Shahri, S. Parris, F. Saki, I. Panahi, and N. Kehtarnavaz

Department of Electrical Engineering, University of Texas at Dallas, USA

ABSTRACT

This paper presents the steps one needs to take in order to run a signal processing algorithm designed in Simulink on the ARM processor of smartphones. The steps are conveyed by transitioning two signal processing application examples from Simulink to smartphone. The application examples involve background noise classification and lane departure detection. Considering that Simulink programming is widely used in signal processing, the approach presented in this paper is of benefit to practicing engineers and signal processing researchers/educators in terms of the process of making Simulink codes or models to run on smartphones.

Index Terms—Simulink to smartphone, running Simulink models on smartphones, background noise classification on smartphone, lane departure detection on smartphone.

1. INTRODUCTION

ARM processors are extensively used in modern smartphones as their main processing engine. In [1] [2], it was shown how to use smartphones as a hardware platform to run signal processing algorithms in real-time. For this purpose, C programming within the software development environment of smartphones was used. This paper presents an alternative programming approach to run signal processing algorithms in real-time on smartphones, in particular on Android smartphones noting that about 80% of all smartphones in use today are Android-based [3].

This approach is based on the newly released Android smartphone Simulink support module by MathWorks [4]. Simulink [5] is a block-based extension of MATLAB that allows systems to be put together in a graphical way referred to as a model. Simulink is widely used for the programming of signal processing algorithms. Thus, a process that easily enables the implementation of a Simulink model on the ARM processor of a smartphone is of interest to smartphone app developers as well as signal processing researchers and educators.

The objective of this paper is thus to show the steps one needs to take in order to run Simulink models on smartphones. To help readers to more easily perform the steps needed for this transition, two signal processing application examples are provided in this paper. These applications consist of background noise classification and lane departure detection.

The rest of the paper is organized as follows. Section 2 describes the software tools used to enable the implementation of Simulink models on an Android smartphone target. Section 3 covers the details of the two Simulink example models involving background noise classification and lane departure detection as well as an added functionality to enable graphical user interfacing which currently is not part of the smartphone support provided by MathWorks. Finally, the actual real-time timing of these implemented applications on an Android smartphone is reported in section 4 with the conclusion appearing in section 5.

2. SMARTPHONE SIMULINK SUPPORT

To enable the deployment of Simulink models onto an Android smartphone target, the following software tools are needed: Samsung Galaxy Android Support package (SGAS package) [4], Android Development Tools Bundle (ADT Bundle) [6], Android Native Development Kit (Android NDK) [7]. Here it is assumed that the reader is familiar with Simulink and the above Android development tools.

The steps one needs to take for a Simulink model to run on an Android smartphone appear in Fig. 1. The first step consists of adding smartphone supporting blocks from the SGAS package to the Simulink model. The SGAS package incorporates a library of Simulink blocks that enables various Android smartphone i/o and sensor devices to be used in Simulink models as either input blocks or output blocks. For example, the Audio Capture block enables frame-based recording of stereo audio signals using the microphones of Android smartphones.

Next, the Simulink model with the supporting blocks from the SGAS package needs to be compiled. When deploying the model to a smartphone target, the SGAS package generates all of the code necessary for the Android application. Although some functionality is provided using Java code, for the most part, the bulk of the signal processing code is implemented in C via the Simulink Coder [8] to automatically generate the C source code describing the Simulink model.

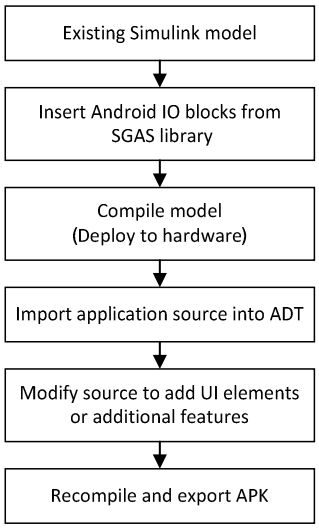


Fig. 1. Simulink to smartphone transition flowchart

When running cooperatively with a computer, the smartphone can communicate wirelessly with the computer to enable data to be displayed within the model or sent to a file. In addition, model parameters can get adjusted in real-time while the application is being run. Alternatively, the model can be deployed as a stand-alone application to a smartphone target. A stand-alone application does not need to be tied to a computer enabling field testing via the smartphone. However, since the SGAS package does not currently possess the support for implementing a user interface for situations that parameters need to be changed or the values of certain variables need to be captured or recorded, this capability has been added as part of this work. This is achieved by incorporating additional features using the Simulink-generated source codes as the starting point. For this purpose, the Simulink compiled model is imported into ADT to allow adding UI (user interface) elements. It is worth mentioning that it is possible to add other capabilities that are currently not offered by the SAGS package. For example, , the function *System.nanoTime()* allows measuring the running time in nanoseconds and the function *getMemoryInfo()* allows measuring the memory consumption. The modified source code is then recompiled in ADT. This generates the application as an APK app file that can be run on the smartphone.

3. SIGNAL PROCESSING APPLICATION EXAMPLES

3.1. Background noise classification

This application example consists of the classification part of a previously developed signal processing pipeline. This example involves automatic classification of different background noise environments for the purpose of tuning the speech enhancement component of cochlear implants according to the classified noise type [9-11]. The pipeline consists of two parallel paths: a speech processing path and a noise classification path, both running in real-time. The speech processing path includes a parameterized noise suppression component which is automatically tuned according to the noise class or type identified by the noise classification path. The background noise classification is written in Simulink, see Fig. 2, and used here as an application example for going from Simulink to smartphone

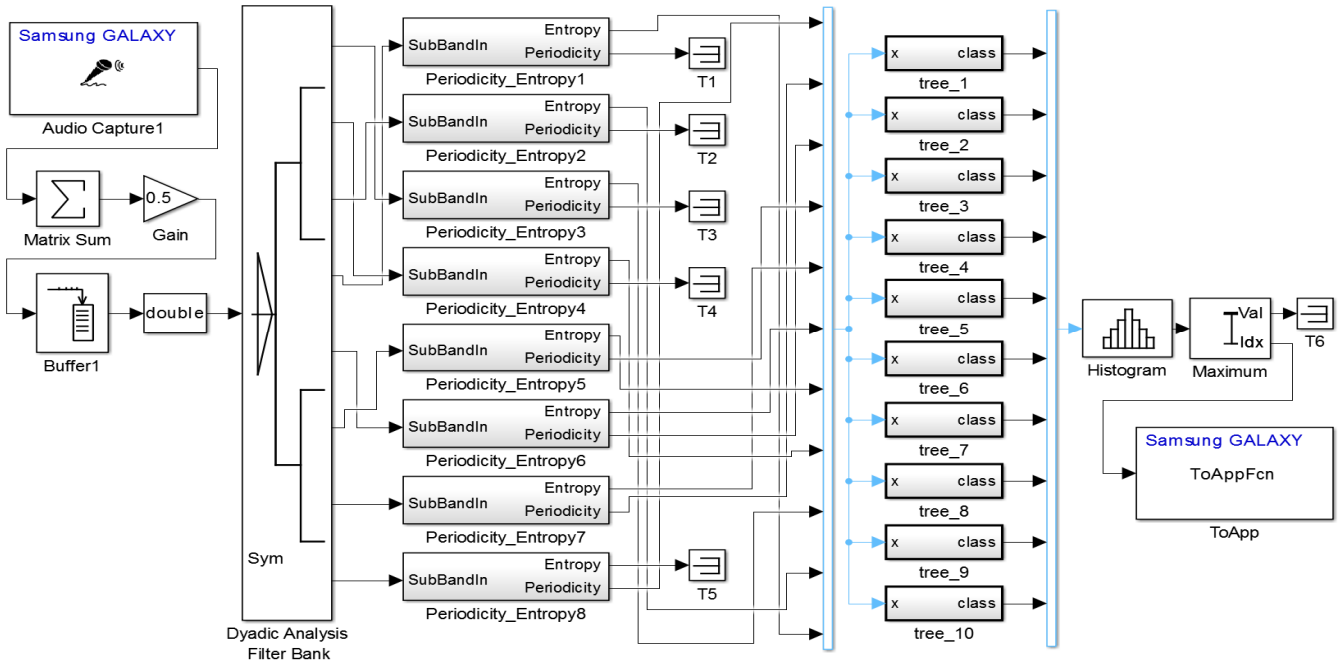


Fig. 2. Simulink model of background noise classication in [9] as an application example

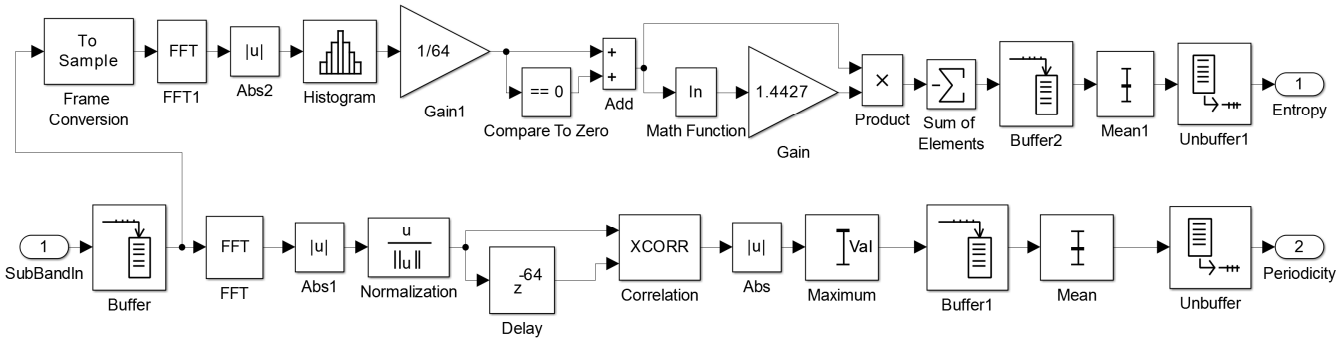


Fig. 3. Feature extraction Simulink block

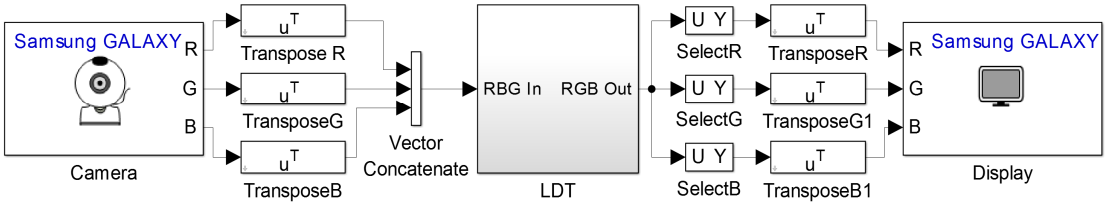


Fig. 4. Simulink model of lane departure detection in [12] as an application example

implementation.

The noise features considered include band periodicity (BP) and band entropy (BE). The random forest (RF) classifier is used to classify noise signal features. The classification decision is made based on the most voted class over all the trees or by majority voting. Details of the features and classifier appear in [9].

As shown in the Simulink model of the noise classification in Fig. 2, the leftmost blocks capture the microphone audio signal. To extract subband features, the noise signals are segmented into one second intervals across 8 subbands. These segments are then divided into forty 25-ms non-overlapping frames. The components of the feature extraction Simulink block are shown in Fig. 3. The band periodicities of the first 7 subbands as well as the band entropies of the first 4 subbands are used to form an 11-dimensional feature vector. Each tree of the RF is programmed as a separate block. The outcomes of all the trees are combined to make a decision on the noise class by allocating the winner to be the class with the highest number of votes.

In order to make the application suitable for smartphone field testing, the functionality of a UI is added to view diagnostic information about the model. To accomplish this, the model is first run as a stand-alone application with the desired outputs routed to a "ToApp" block in the model. This allows gathering all of the desired variables in one place as the computer assigned names are difficult to interpret. Once the application source code is generated, it is then imported into the ADT and the tools are configured so that the application is compiled independently from Simulink. Afterwards it is a matter of coding the additional desired functionality. This involves adding a console-like textual interface. Within the C code of the application, the timing functionality is added to benchmark the run time performance of the model. Furthermore, the functions necessary to call the Java code to display diagnostic messages in the user interface are added.

3.2. Lane departure detection

The second application example involves the Simulink model of lane departure detection provided by MathWorks [12]. This example includes detecting and tracking road lane markers in a video sequence by using Hough transform and Kalman filtering blocks. The original Simulink model reads video frames from a file and displays the outcome in the form of an overlay on the video frames. This overlay consists of line markers which show the boundaries of a lane together with its surface. To make the model run on an Android smartphone, it is modified here to receive the video frames from the smartphone camera and show the lane boundaries on the smartphone screen. The modified model is shown in Fig. 4 where the LDT block denotes the fixed-point Simulink implementation of the lane departure detection model appearing in [12].

4. REAL-TIME PROCESSING OUTCOME

In this section, the actual smartphone implementation of the above two application examples of background noise classification and lane departure detection is reported.

The implemented background noise classifier is designed for three common noise environments: babble noise (e.g., restaurant, mall), road noise, and machinery noise. These noise classes are encountered often on a daily

basis. Sample spectrograms of these three noise environments are shown in Fig. 5. As can be seen from these spectrograms, the noise characteristics appear different which are captured by the subband features.

The Simulink model processes audio files or audio signals at the sampling rate of 44.1 kHz. The training was done on a computer and only the testing or actual operation was carried out on the smartphone. The classification accuracy was similar to that which was reported in [9], i.e. about 99% with majority voting over 5 seconds. The developed model together with the UI was deployed on a Samsung Galaxy S4 smartphone. Fig. 6(a) shows a picture of the smartphone with the app running in a location where the background noise was babble. The items on the UI included the detected class as well as the running time for one second of audio data. As can be seen from Fig. 6, the processing time was a tiny fraction of this time, about 0.4 ms, which easily allowed the model to run in real-time. The app of this Simulink model running in real-time on an Android smartphone can be downloaded from *http://www.utdallas.edu/~kehtar/SimulinkApp1.apk*.

The lane departure detection model was also deployed on a Samsung Galaxy S4, similar to the first example. The camera block allowed selecting the image resolution from a list of predefined resolutions as well as a desired frame rate. The video capture frame rate was set to 30 fps. However, in order to meet a real-time processing throughput on the smartphone, the image resolution was set to 320×240. Fig. 6(b) shows a screenshot of this application example. The app of this Simulink model running in real-time on an Android smartphone can be downloaded from *http://www.utdallas.edu/~kehtar/SimulinkApp2.apk*.

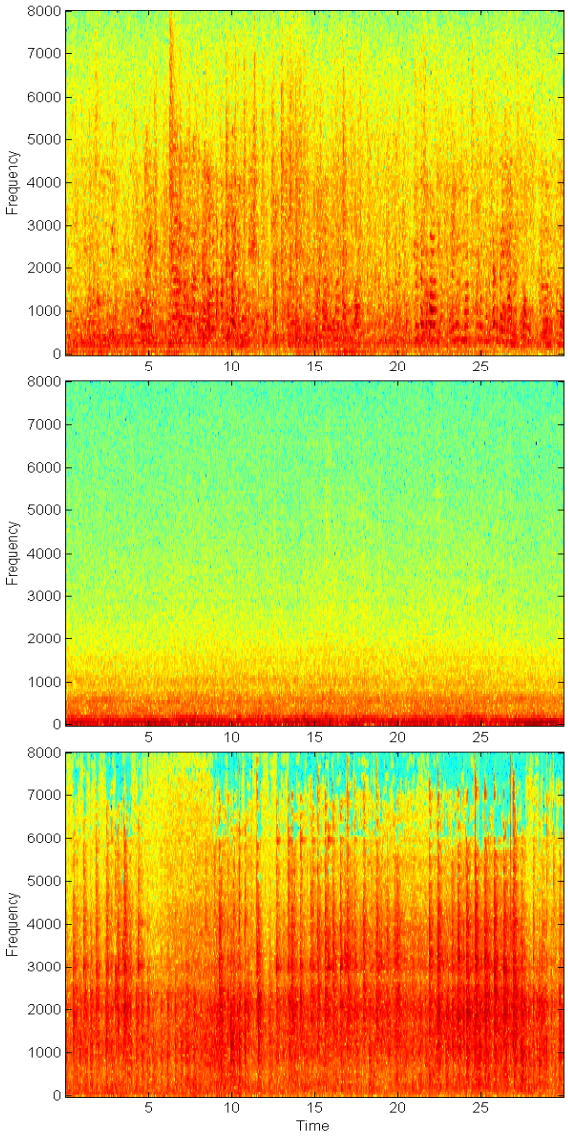
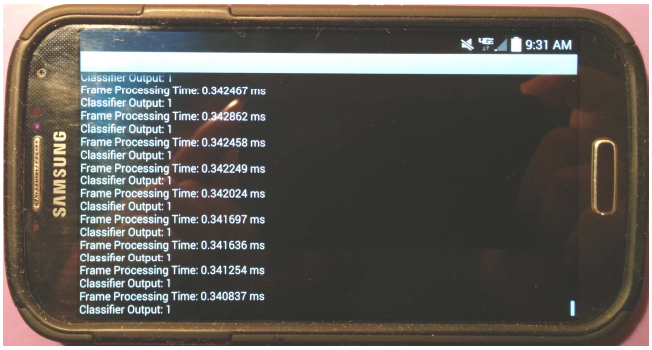


Fig. 5. Spectrograms of three sample noise signals (x-axis in seconds and y-axis in Hz): babble (top), road (middle), machinery (bottom)

5. CONCLUSION

This paper has presented the steps one needs to take to run Simulink models on Android-based smartphones based on the support package provided by MathWorks. Two application examples consisting of background noise classification and lane departure detection are transitioned from their Simulink models to actual smartphone implementation exhibiting that these steps do not require one to deal with C/Java and models can be directly transferred from Simulink to smartphone.



(a)

(b)

Fig. 6. Screenshot of (a) background noise classification, (b) lane departure detection Simulink models running on an Android smartphone

6. REFERENCES

[1] N. Kehtarnavaz and S. Parris, "Teaching Digital Signal Processing on Smartphones: A Mobile DSP Laboratory," *http://www.icassp2014.org/show_and_tell.html*, Italy, May 2014.

[2] S. Parris, M. Torlak, and N. Kehtarnavaz, "Real-time implementation of cochlear implant speech processing pipeline on smartphones," *Proc. of IEEE International Conference Engineering in Medicine and Biology (EMBC)*, Chicago, August 2014.

[3] *http://www.gartner.com/newsroom/id/2665715*

[4] *http://www.mathworks.com/hardware-support/android-programming-simulink.html*

[5] *http://www.mathworks.com/products/simulink*

[6] *http://developer.android.com/sdk/index.html*

[7] *http://developer.android.com/tools/sdk/ndk/index.html*

[8] *http://www.mathworks.com/products/simulink-coder*

[9] F. Saki and N. Kehtarnavaz, "Background noise classification using random forest tree classifier for cochlear implant applications," *Proc. of IEEE ICASSP*, pp. 3591-3595, Italy, May 2014.

[10] V. Gopalakrishna, N. Kehtarnavaz, T. Mirzahasanloo, and P. Loizou, "Real-time automatic tuning of noise suppression algorithms for cochlear implant applications," *IEEE Transactions on Biomedical Engineering*, vol. 59, pp. 1691-1700, 2012.

[11] T. Mirzahasanloo and N. Kehtarnavaz, "Real-time dual-microphone noise classification for environment-adaptive pipelines of cochlear implants," *Proc. of IEEE International Conference Engineering in Medicine and Biology (EMBC)*, pp. 5287-8290, July 2013.

[12] *http://www.mathworks.com/help/vision/examples/lane-departure-warning-system-1.html*