Digital Audio Filter for Small Unmanned Aircraft

Steven Krukowski and Marcus Hammond

EE 264 Digital Signal Processing Final Project

3/19/2015

Introduction

One of the largest uses for the emerging small unmanned aircraft market is video capture. Technology advanced in small high-bandwidth antennas has allowed for simpler first person video (FPV) flying, where the pilot flies the small aircraft as if they were in the cockpit. While video is very useful, the accompanying audio is often useless due to several sources of background noise which overpower any desired audio sources. The majority of this noise for a quadcopter comes from the motors and the propellers. One of the goals of this project was to characterize and digitally filtering this noise would in order to enhance a future FPV flying experience.

Another challenge in the flying of small unmanned aircraft is obstacle avoidance with a limited field of view. Especially when flying FPV with a quadcopter, one might desire to move in a direction which the camera is not facing (i.e. pan-out or pan-left or right); however a wall collision can result in costly damage to both the aircraft and the property. Instead of filtering out the background rotor noise, the goal of the second part of the project would be to use the background motor noise for wall detection. Using a stereo microphone, the aircraft's directional proximity to a wall will be detected using the echo effect off the wall. This could be used to give an FPV operator a heads up similar to a back-up detector on a car.

Timeline

Table 1 shows both the intended timeline and the actual timeline of work actually into this project. Due to an extension of the deadline, nothing was originally planned for the final week of the project.

Table 1: Project Timeline

Week	**Goal**	**Actual**
Week 1: 2/16-2/22	Digital Noise characterization and analysis	Recorded Digital Noise characterization (Matlab)
Week 2: 2/23-3/01	Background noise filter implementation	Live Digital Noise characterization (DSP) and
Week 3: 3/02-3/08	Wall detection proximity implementation	Noise filter design and wall proximity characterization
Week 4: 3/09-3/13	Flight Testing, debugging, overflow, report writing	Wall proximity filter design and demo preparation
Week 5: 3/14-3/19	N/A	Wall proximity detection tuning and report writing

Implementation

Implementation of the project happened in several steps. The following sections will describe the vehicle test-bed, vehicle noise characterization and analysis, the audio filter design, and the wall proximity filter and detection design.

Vehicle Test-Bed

A picture of our vehicle test-bed is found in Figure 1. The two given microphones were plugged into the DSP board using a stereo input splitter. The plastic container was designed to be mounted with the microphone rod off the bottom of the quadcopter on a vibration damped mounting plate. The DSP shield could be powered by an external cell phone battery charger (not pictured). With the exception of the sound characterization, most of the testing was done with the rotor blades off for safety concerns. During characterization testing a computer would be connected to the DSP board via USB and audio output. During implementation testing, a pair of noise cancelling headphones was connected to the DSP audio out.

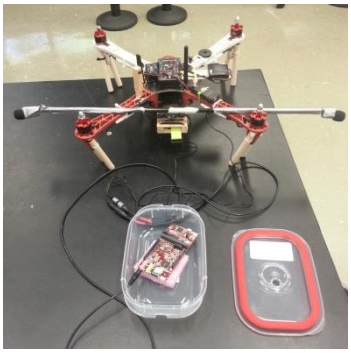


Figure 1 Test-bed Configuration

Noise Characterization

The rotor noise was characterized and compared to the ambient noise. Figure 2 shows a spectrogram of those results. During this test, the motors were swept through different speeds. From this characterization, we noticed two important points. The first was that the strongest frequencies of vehicle noise were at low frequencies and high frequencies. This helped in the design of the audio filter because the strongest noise came below 300 Hz and around 8000Hz. The human speech fundamental frequencies are below 300 Hz, but the harmonics are above 300 Hz. This meant that we could filter out a large portion of the rotor blade noise while still picking up human speech from its harmonics. The second point was the existence of a constant, narrow band of high frequency noise that comes from the pulsing of the motors. It was decided that this constant sound could be used for wall detection.

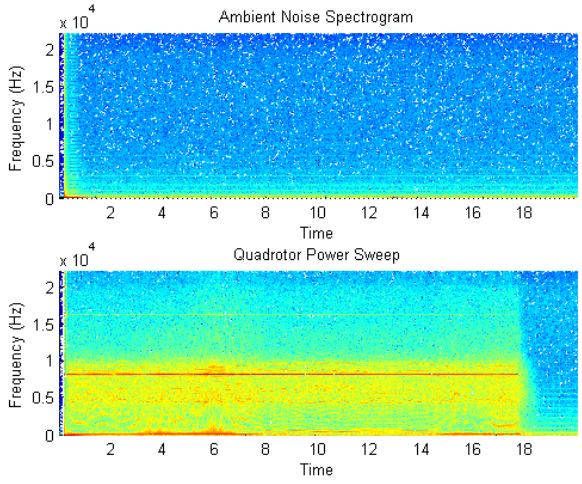
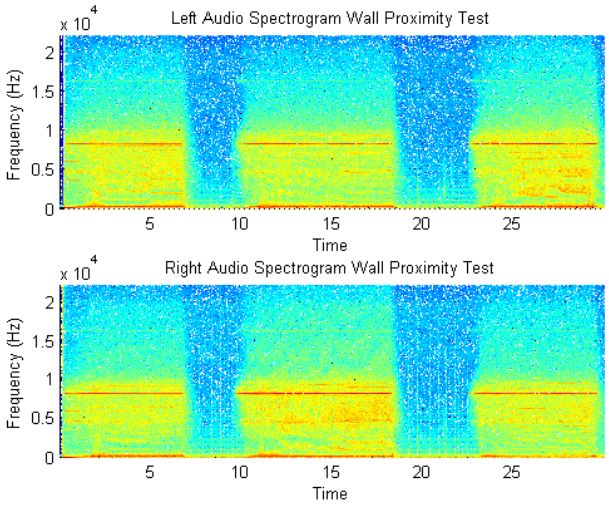


Figure 2 Vehicle Noise Characterization

The next step was to characterize a difference in the audio inputs when the vehicle was placed near a wall. The results of this test are found in Figure 3. This test was split into three segments. The motors were turned on not near a wall (0-6s), then with one side close to a wall (10-18s), and finally with the other side close to a wall (24-29s), with the motors turned off in between tests. Again, there is a distinct frequency band around 8000 Hz, for all three tests. Further testing was performed to see if the power at this frequency was affected by proximity to the wall. The results from this testing is show in Figure 3. The sound was filtered using an FIR filter discussed in future sections and the average power (in a given interval) was calculated for the left and the right audio input separately (this was filtered using an IIR). In the ideal case, the left and the right audio would have close to equal power when not near a wall; however, the given microphones were not equal and it can be seen the left microphone (green line) was more sensitive than the right microphone (blue line). The important point of this characterization was that the differential power between the two was dependent on whether the microphone was near the wall and could be used to detect proximity to the wall



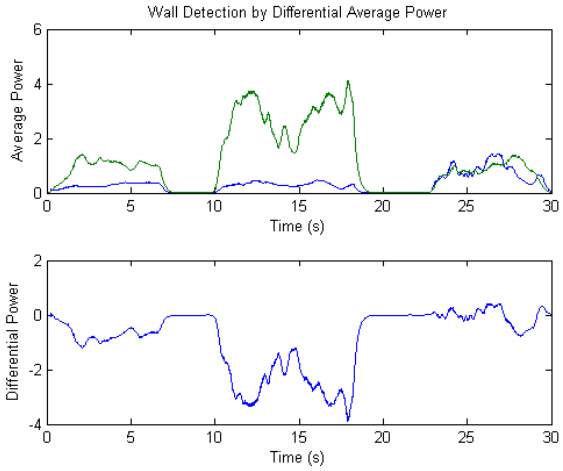


Figure 3 Wall Detection Characterization

Audio-Out Filter Design

To filter out the audio noise, a Park McClelland FIR filter was designed in Matlab. The specifications used are found in Table 2. A sampling frequency of 44100 Hz was used for the implementation of this part.

Table 2: Audio Output Filter Specifications

Band	Frequencies	Magnitudes (abs)	Ripple (abs)
Stop Band 1	0-300Hz	0	0.01
Pass Band	400-6800Hz	1	0.1
Stop Band 2	7000-Fs/2	0	0.01

The frequency and the impulse response of the resulting filter can be found in Figure 4. The ripple was kept to a low magnitude in the stop bands due to the high noise of the rotors while the ripple in the pass band was not as important. The end of the first stop band was chosen to be 300 Hz because the major of the low frequency rotor noise was below 290 Hz. Ideally the pass band would have started closer to 300 Hz, but this caused very high order filter results. The second stop band started at 7000 Hz because this was well above the range of human speech harmonics, but well below the 8000Hz frequency of the high frequency motor noise. Different spacing was experimented with, and many caused high magnitudes in between the frequency bands or very high order filters. The resulting filter had an order of 551, which seems practically implementable. A Parks McClelland FIR was chosen over an IIR because the resulting FIR has linear phase, with a constant group delay of 275.5. For the sampling rate of the DSP, this was considered an acceptable lag and because the group delay was constant, there would not be distortion in the audio output. The audio out filter was only implemented and tested in Matlab so more time could be spent on the second objective of this project.

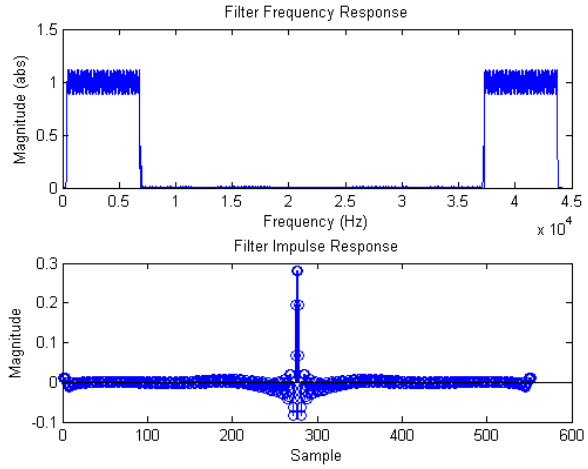


Figure 4 Audio Output Filter Response

Wall Proximity Filter and Detection

As stated in the noise characterization, the high frequency motor noise was used in the detection of proximity to a wall. Again a Parks-McClelland FIR filter was designed in Matlab with the specifications found in Table 3.

Table 3: Wall Detection Filter Specifications

Band #	**Frequencies**	**Magnitudes (abs)**	**Ripple (abs)**
Stop Band 1	0-7100Hz	0	0.001
Pass Band	7600-8600Hz	1	0.01
Stop Band 2	9100-Fs/2	0	0.001

The pass band was chosen narrowly around 8000 Hz motor noise and the resulting filter frequency response and impulse response is found in Figure 5. This filter was implemented on both the left and right audio input on the DSP shield using the software-based FIR functions. After the audio is filtered, the short time average power is calculated for both the left and the right audio. The average power on each channel was still noisy so a first order low-pass IIR was

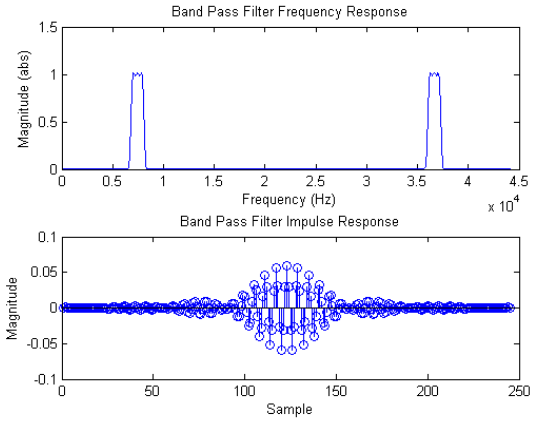


Figure 5 Wall Detection Filter Response

implemented on the average power in order to smooth the response. Finally the difference in average power of the left and the right was calculated. If the average power difference was greater than a certain value, it was declared near a wall on the one side. Alternative if the average power difference was less than a certain value, then it was declared to be close to a wall on the other side. If the average power difference were between these two values, then it was declared to be not near a wall on either side. Several implementations were explored on how to declare being near a wall, including using the on-board LEDs or the OLED screen. The method that worked best was outputting the high frequency tone, which was used to detect the wall, on the appropriate audio output channel if a wall was detected on that side.

Results

The audio output filter was implemented and tested in Matlab. The results of the audio output can be seen in Figure 6. The noise at the low and high frequencies is greatly reduced in the lower plot. This was consistent with the audio results. Besides an overall decrease in rotor volume, the high frequency (8000 Hz) "whine" is significantly reduced. This filter was not implemented an tested on the DSP shield, leaving room for future work.

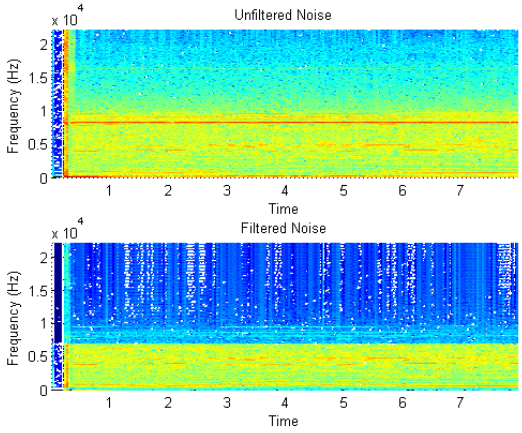


Figure 6 Audio Filter Results

The wall proximity algorithm was tuned and implemented on the DSP board. Tuning consisted of saving the time-average right power, left power, and differential power outputs (for wall on left, wall on right, and no wall) and outputting them to Matlab on command. From these the thresholds described in the implementation were chosen. The wall proximity was then tested by having one person using headphones but not seeing whether the vehicle was placed near a wall. They would raise their corresponding hand if they heard the tone in that ear. Results from the wall proximity detection were mixed, depending a large part on the threshold tuning. The microphones given were not equal in sensitivity and as a result, the wall detection worked better on the one side than the other. Due to continuing work on this part, tests were not quantified, but instead only qualitative results were observed.

Class Concepts Used

Much of the knowledge used during the class was used on the project but the course subject areas that were found most useful for this project were:

- Digital Filter Design and Implementation (Lab 3 and 4)
- FFT and Spectrum Analysis using the DFT
- Time-dependent spectrum analysis (Spectrograms).

Future Work

There is still plenty of interesting work to be done on both goals of our project. The hardware was developed, but the audio output filter was never implemented on the DSP board or tested in flight. Consideration will be given to using some of the board's hardware accelerators in order to assist in the audio filtering. Also, both of us would like to continue work on the audio wall detection based on promising results from this project. Possible future changes to the wall detection include using higher quality microphones, better tuning methods for threshold values, and use of hysteresis in the detection. We both would like to keep our DSP boards in order to further develop these ideas.

Appendix

We were unfamiliar with DOXYGEN and were unable to get it to work. Below are the function descriptions for the DSP code. Full code is also attached

/** \brief Setup function

Allocate memory for input/output arrays.

Initialize OLED display and Audio module.

*/

/** \brief Main application loop

Read serial commands from matlab to control recording

*/

/**\brief Proess Command Function

\param command is an integer correspond to the serial command used in tuning

0: Start recording

1: Stop record

2: Display Power differential on the OLED

*/

/// \brief Main processing function

/**

\param inputLeft is a pointer to the input data array for the left channel

\param inputRight is a pointer to the input data array for the right channel

\param outputLeft is a pointer to the output data array for the left channel

\param outputRight is a pointer to the output data array for the right channel

*/

```
/** \brief Set Tuning Parameters
  \param leftPower the filtered left audio power
  \param rightPower the filtered right audio power
  Sets the average, max, min of leftPower, rightPower, and dPower along with the number of samples
  taken to be sent over serial to Matlab when tuning the thresholds.  Called during processData
*/

/** \brief DMA Interrupt Service Routine
  Manage ping-pong input/output buffers.
  Calls processData() on input/output buffers.
*/
```