April 4th 2016

P2: Final Project

CS 489 001: Computational Audio

Sarah Fraser (sm2frase : 20458408)

Final Project Description:

I have created an application (in MATLAB) that works as a POC for a more general application. The general app takes in an audio file (.wav) of a bird call and tries to match the call to a database of known birds. My POC does the same thing but at a smaller scale by only having 10 known birds in the database and trying to match only a select group of bird calls.

To run the app:

Open MATLAB

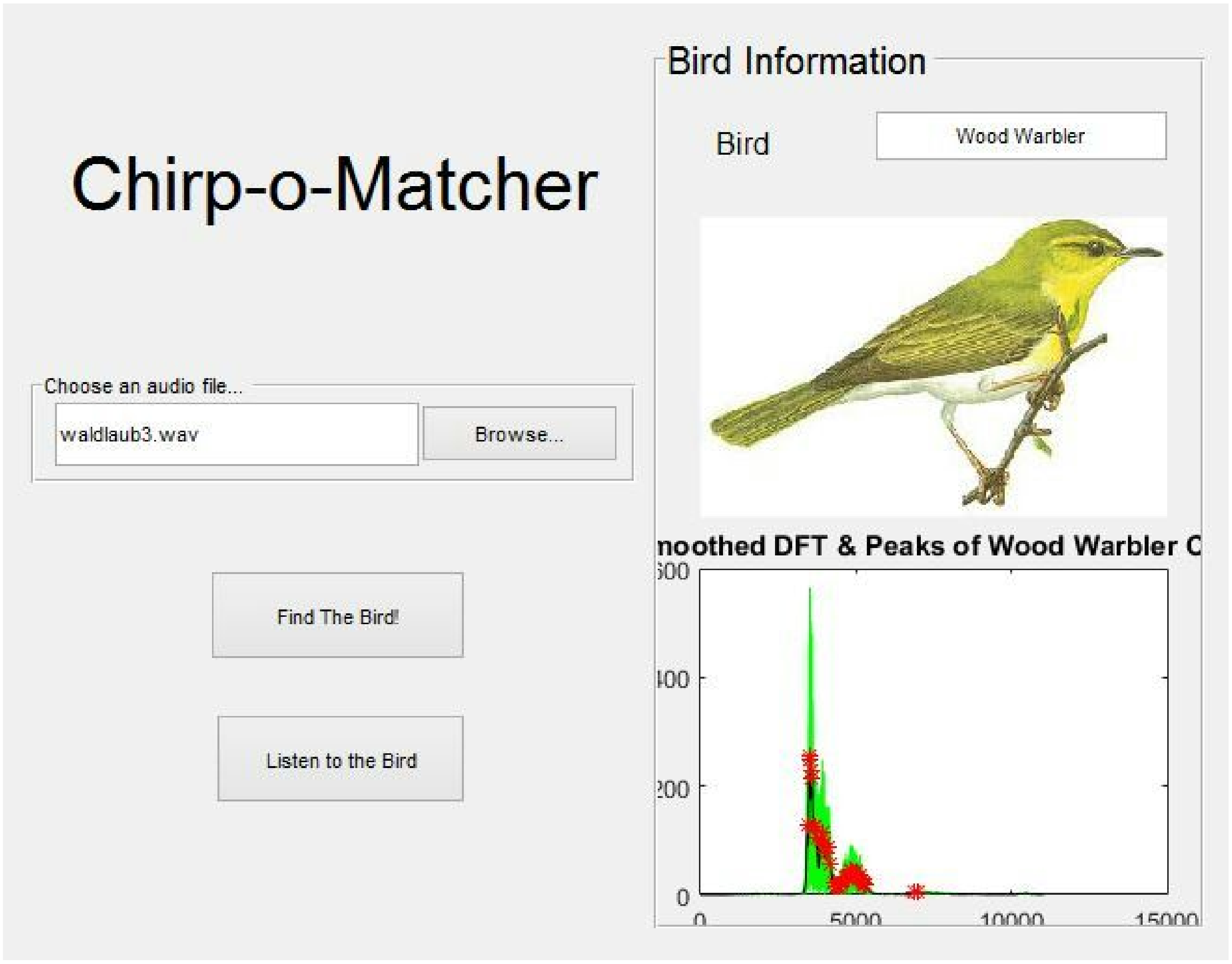
```
>> run chirp_o_matcher.m
```

Files to use for evaluation/testing:

amsel2.wav (Blackbird)
blackter.wav (Black Tern)
blm2.wav (Blue Tit)
bsp3.wav (Great Spotted Woodpecker)
corncrak2.wav (Corncrake)
Gambeli m.wav (Mountain Chickadee)
heckenbr.wav (Hedge Sparrow)
mocking.wav (Northern Mockingbird)
waldlaub3.wav (Wood Warbler)
wendehal.wav (Wryneck)

App Overview

The GUI for my app is fairly simple. There is a file upload button that allows you to choose which audio file you wish to analyze. There is also the option to enter the filename mainly. To then determine which bird the bird call came from, one simply has to press the “Find the bird!” Button. Then, on the right half of the app window, the bird name should appear, along with a picture of the bird and the DFT (smoothed & not smoothed) with peaks.



My Fingerprint Algorithm
As indicated in my P0 and P1, my plan was to implement some sort of audio fingerprinting and then try to match the fingerprint of the unknown bird to a known bird fingerprint in the database. Initially, the proposed algorithm I wanted to use was taking the DFT and condensing it into a hash with focus on the time domain so audio partialsédifferent recordings could match the same audio file. However, after reflection as described in my P1 report, this would have been much too complex for me to do in the time provided since bird calls have a large amount of variance. Therefore, I had to change my fingerprint algorithm. My new plan was to use the DFT of small sections of the audio but after some comments from Prof. Mann, I concluded that using the overall DFT and smoothing it would work in my favor.

The fingerprinting algorithm:

1) Compute DFT of audio from .wav file
2) Smooth the DFT with a window width of 10 Hz

3) Use the smooth DFT to compute the peaks
4) Eliminate any peaks below 5 Hz
5) Divide the frequency range (0Hz-15000Hz) into 100Hz bins (ie. bin #1 = 0 - 100 Hz, bin #2 = 100 - 200) etc
6) For each frequency bin, find the highest peak (the frequency with the most value) using the computed peaks
7) The fingerprint was then set to be the the highest peak for each bin (with a length of 150 values)

Some comments:

- I chose to use the smoothed DFT (as suggested by Prof. Mann) since it was much easier to work with and provided an easier comparison. I also chose to have a window width of 10 Hz because after trying a variety of widths, I found this one best represented the data
- I chose to eliminate any peaks below 5 Hz because after evaluating the smoothed DFT for each databased bird call, I found that all of the significant peaks were 5+ Hz. Thus eliminating the lower, insignificant peaks allowed for a more streamlined analysis.
- I decided on the frequency range of 0Hz - 15000 Hz because the DFTs of the databased bird calls varied quite a bit in frequency. Some birds had a broad range while others clumped near the lower part of the spectrum while others clumped near the higher part. Therefore, in order to do a comparison of all of the bird calls, I had to come up with a range that encompassed all of the calls.
- I choose to use bins and find the max peak instead of just comparing point by point for a few reasons:
    - I wanted to ensure that if the audio file was “off” by a bit and did not match up with the database file, I could still get a reasonable match
    - I also wanted to ensure that I could do a comparison of every single databased audio file. None of my audio clips are of the same length, and thus do not have the same number of data points. By using a predetermined bin size that was consistent for all audio DFTS, I was able to compare everything equally.
- My bin choice of 100 Hz seemed reasonable to me since it provides enough precision to get an accurate match but was not too precise that it was the same just comparing the two DFTs

Matching Algorithm:

1) For each bin in the unknown bird call’s fingerprint:

    a) Compare the difference between the peak value for each databased bird
    b) Find the smallest difference
    c) Add a “match” for the databased bird that had the smallest difference
2) After each bin has been analyzed, find the databased bird that had the most “matches”. This bird should be the unknown bird.

Reflection of Revised P1 Goals:

1. Choose 10 unique bird call recordings of around the same length to use in POC
2. Implement a basic algorithm that is able to create a fingerprint of a sound with no concern to the time domain or factors and then use it to try and match two of the same bird calls together. Essentially, use the DFT of one bird call and try to match its DFT (within a reasonable error bound) to another DFT of the same bird call recording.
3. Add DFT smoothing and try to match the peaks of the DFTs vs every DFT datapoint
4. Do goal 3 but with more concern to error bounds and not a perfect matching
5. ~~Do goal 2 but with focus on the time domain~~
6. Finalize the fingerprinting algorithm and create one bird call fingerprint to use for comparison on all other inputs from the 10 chosen bird calls
7. Create a program that takes any bird call input from the 10 chosen bird calls and uses the fingerprinting algo to try and match it to the stored fingerprinted bird call. Essentially the program will be “Is this bird call a (type of bird) call?”
8. Create the database with multiple bird calls using different bird call audio inputs from the chosen 10
9. Implement an efficient search algorithm to match a fingerprint to the database
10. Entire project has been completed
11. Create a usable GUI

Overall, I was able to meet almost all of the goals I set out for myself, as well as then some. I had to add a few revisions here and there due to my change in the fingerprinting algorithm, but overall after reevaluating my goals in P1 I was able to meet them.

Reflection on Project Impact

Overall, I think that this project has really helped me learn more about audio analysis and comparison. Not only did it open my eyes to just how complex seemingly simple sounds like bird calls can be, but also that matching them takes a lot more work than just looking the the raw data. I have found that I have a new respect for anyone who has created a sound matching algorithm.

Also, I think that this project has really helped further my understanding of concepts learned in class. By doing more research into things like DFT and smoothing, I was able to get an even better understanding of the topics we touched on in class.

Files Included in Project Package:

- P0
- P1
- 10 bird call .wav files
- Basic audio signal of each bird call
- DFT of each bird call (no smoothing)
- DFT comparison of smoothing/ no smoothing as well as peaks
- fingerprint_call.m (MATLAB function used to create a fingerprint for a .wav file)
- Match_fingerprints.m (MATLAB function used to match a fingerprint to a fingerprint in the database)
- Chirp_o_matcher.m (MATLAB APP GUI)
- Chirp_o_matcher.fig (MATLAB GUI file)
- Fingerprints of 10 birds