Overlapping Voice Separation

Team 9 – daea

Daniel Payan, Alex Farfan, Evie Holyoake, Alex Brown

Background / Rationale

- It is hard to achieve effective communication between the pilot and ground control regarding the pilot's biometric conditions
- Want to utilize the audio streams of the flight for retrospective analysis of how external factors impact pilot health
- Currently, interferences such as machine whir, wind shear, and overlapping voices complicate the analysis process
- The audio interferences present in the audio stream can lead to a considerable amount of manual effort to evaluate pilot health and performance
- A system that could isolate speakers using predetermined audio profiles and separate overlapping audio segments would aid in-flight analysis

Objectives

- Implement a voice identification system that runs live on an embedded environment, the Google Coral Dev Board
- Implement an audio filter that removes non-vocal sounds from an audio stream in a live, embedded environment
- Implement a voice separator that splits overlapping speech into separate audio streams
- Implement a design to convert a qualitative audio stream to a quantitative digital vector by creating a Mel Frequency Cepstral Coefficients (MFCC) vector

Gantt Chart


Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
Phase 1 - Setup
Phase 2A - Noise Filtering
Phase 2B - Mask Generation
Phase 3 - Voice Seperation
Phase 4 - Iterative Improvement
BFHP
FP
BH
BFHP
BFHP
B = Alex Brown
F = Alex Farfan
H = Evie Holyoake
P = Daniel Payan

Gantt Chart Phase 1

- ✓Set up workspace/environment
- ✓Begin implementing initial filters
- ✓Use MATLAB to set up FIR/CIC filters to familiarize ourselves with noise reduction implementations
- ✓Set up/Initialize Coral Board
- ✓Research further into quantitative analysis system performance

Gantt Chart 2A – Noise Filtering

✓ Attempt to implement separation of background noise from voices

✓ Separate background noise from a single voice

o Proceed to separate background noise from multiple voices

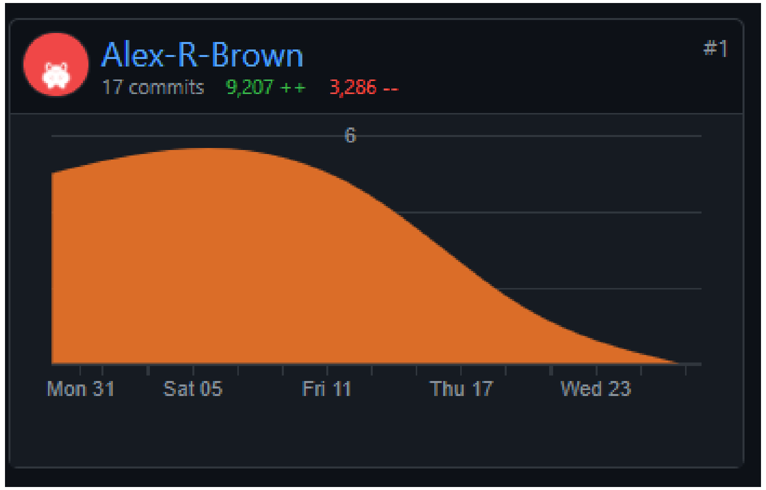
o Proceed to separate background noise from stacked voices

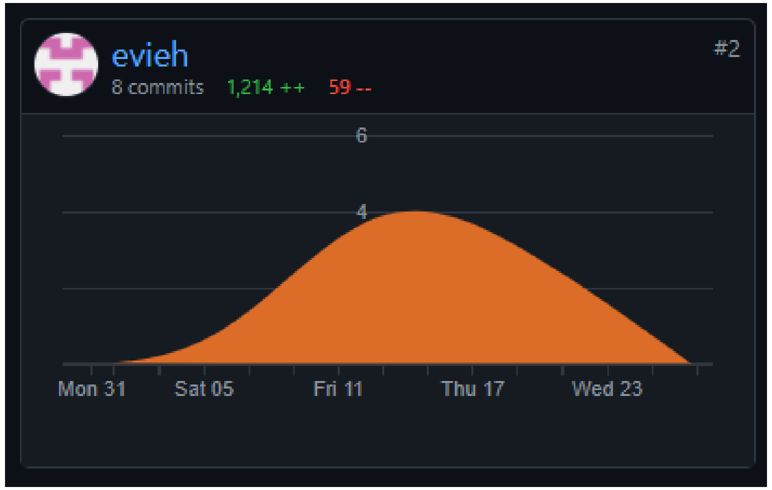
o Implement MFCC feature vector production through MATLAB

o Output MFCC feature vectors to file (for later use by Python3 code)

Gantt Chart 2B – Mask Generation

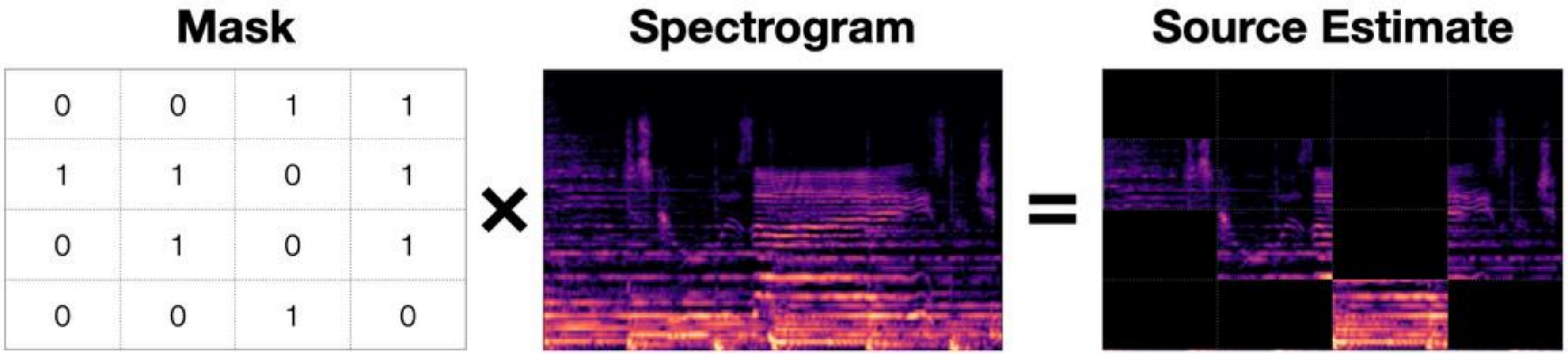
- ✓ Develop infrastructure for interfacing with potential audio clip databases
  - Automated organization and preparation of audio
  - Automated randomized overlay generation
- ✓ Implement MFCC feature vector production through Python3 (utilizing the Librosa library)
  - Capability to control MFCC vector dimensionality
  - Customizable window and step size (~93ms and ~23ms respectively)
- ✓ Input MFCC feature vectors from file
- ✓ Visualize MFCC time-domain data





Gantt Chart 2B – Mask Generation

✓ Research into mask generation based on existing in generating masks in the Time Frequency spectrogram domain



- Setup system to create an "enrollment profile" of audio clips for individuals from database sources
  - We decided to prioritize separation functionality rather than user experience

Gantt Chart 2B – Mask Generation

✓ Implement a mask generation machine learning algorithm
  - MFCC input to output works, but it is a primitive attempt and we are exploring new ways to accomplish this
  - Short Term Fourier Transform (STFT) based off a research paper that focused on source separation in the ideal case (when the knowledge about each source is available)
    - See the mathematical formula to the right

$$y(t) = x(t) + n(t).$$

$$S_y(\tau,\ f) = \sum_{t=0}^{T-1} y(t)g(t-\tau)\exp(\frac{-i2\pi\ ft}{T})$$

$$P_{\cdot}(\tau,\ f) = |S_{\cdot}(\tau,\ f)|^2$$

$$R(k) = \frac{P_x(k)}{P_x(k) + P_n(k)}$$

$$\widehat{S}_x(k) = R(k)S_y(k)$$

$$\hat{x}(t) = \frac{1}{T}\sum_{\tau=0}^{T-1} g(t-\tau)\sum_{f=0}^{T-1} \widehat{S}_x(\tau,\ f)\exp\left(\frac{i2\pi\ ft}{T}\right)$$

$$L(\hat{x},\ x) = \sum_{t=0}^{T-1}[\hat{x}(t) - x(t)]^2 = \sum_{t=0}^{T-1} r(t)^2 = \frac{1}{T}\sum_{\tau=0}^{T-1}\sum_{f=0}^{T-1}\left|\widehat{S}_x(\tau,\ f) - S_x(\tau,\ f)\right|^2$$

Gantt Chart 3 – Voice Separation

- o Continue training mask generation model
- o Use the MFCC feature masks to attempt to separate a known speaker from other speakers
- o Experimentally determine if MFCC feature mask multipliers are sufficient for attenuating non-target voices
- o Experimentally determine if a voice separation model is needed
- ✓Reconstruct audio streams based on MFCC feature vectors with Python3 (utilizing the Librosa library)
- o Implement the noise filtering algorithm on the Google Coral board, testing its performance

Addressing the Deficiency

- Datasets were not provided, a large amount of time was spent finding usable datasets for both filtering and noise separation
- We are working on making sure the progress made is parallel to what's expected
- Ensuring that our sponsor is made aware of blockers and updates

Voice Separation Implementation

- After our previous work in the MFCC and STFT domains fell short of expectations, we are investigating further into the ConvTasNet implementation
- We our basing the next slides off of a pre-trained model from a voice separation online community
- Model only separates two unknown speakers
- Highly complex model architecture with over 23 blocks each containing multiple layers; for comparison ours has 6 rudimentary layers

Voice Separation Demo

- Most popular implementation of the model architecture published by ConvTasNet
- Although this is good, it has some issues
- Going forward our goal is to more closely follow this model architecture, but make it for a specific person.
- This reduction in variability would hopefully yield in greater separation performance

Original Audio

First Stream

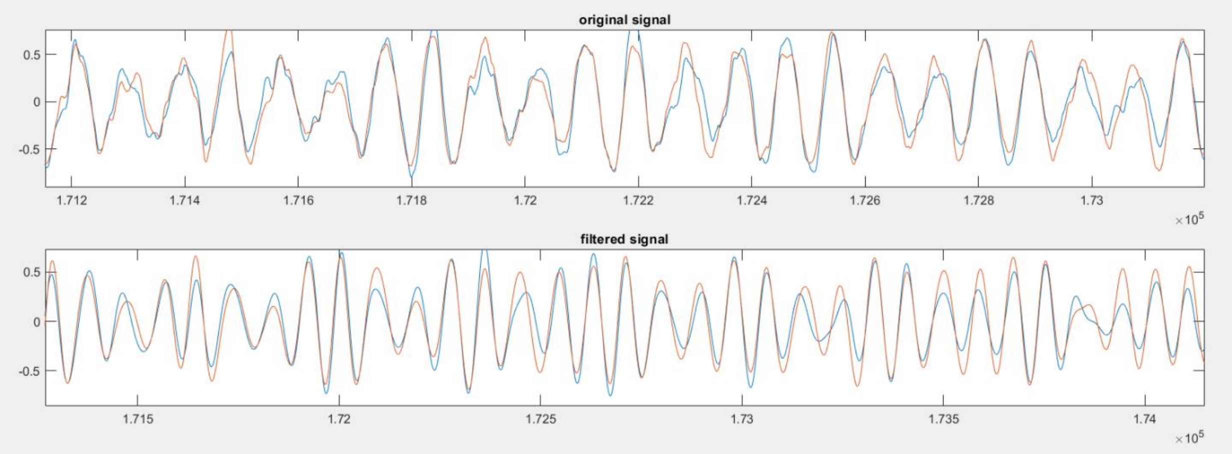
Second Stream

Future Directions for Voice Separation

- Speaker specific models
  - Audio will go through n models that will separate a trained audio profile vs everything else
- Unknown speaker models
  - Audio will go through one model built for a determined number of speakers
  - Multiple models will be built for each circumstance

Filter Implementation

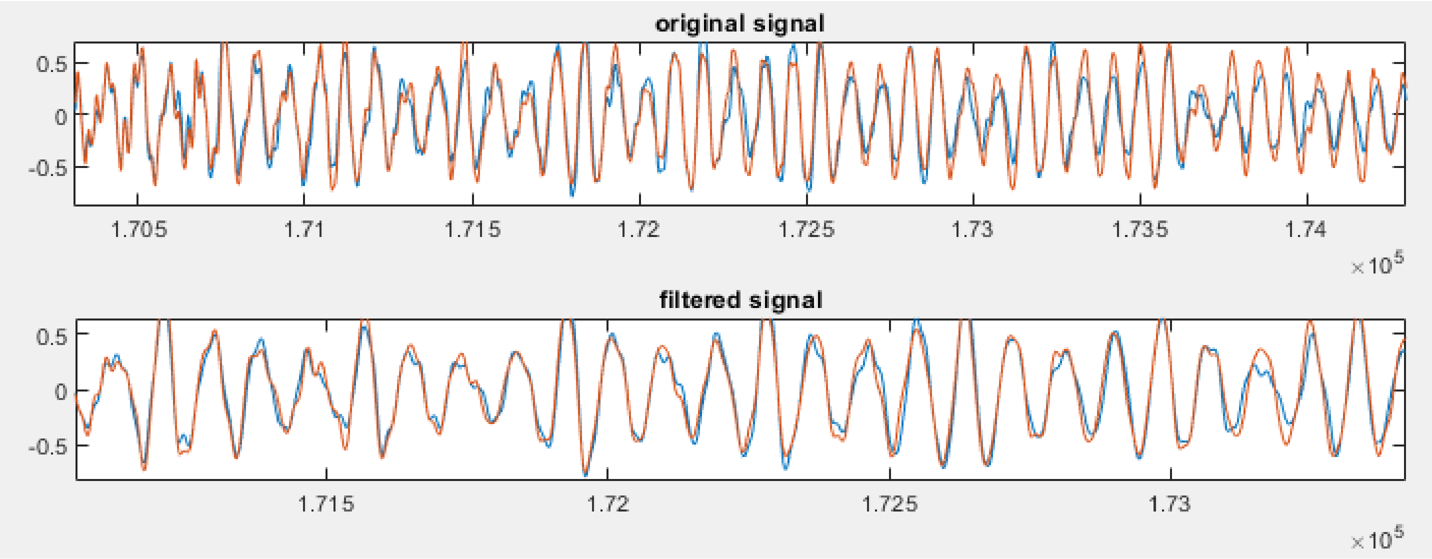
- First implemented a low-pass filter



  - Signal appears to look a little cleaner, but did not sound clean at all because we were passing the low frequency

Filter Implementation

- Passband filter is what we are currently working on (signal doesn't look as clean)



- Next steps: Notch(?), CIC followed by FIR(?)

Filter Demonstration

- As of right now our best model for filter is our passband filter. The passband is currently from 500 hz – 2500 hz and we are messing with the passband range to see what gives best results
- Original Audio:
- Filtered Audio: