Speech Recognition for Air Traffic Control

Faculty advisors:

Jianhua Liu – EECS
Andrew Schneider – Flight Training

Technical support:

Aaron Van De Brook – EECS

Background Information

- Automatic speech recognition (ASR) has long been a very popular research area.
- ASR has many different application, such as personal assistance:
  - Amazon Alexa
  - Apple Siri
  - Google Assistant
  - Microsoft Cortana
  - Samsung Bixby
- While the above personal assistance apps can perform well in normal ASR applications, ASR systems perform less-than-acceptable in applications such as air traffic control (ATC), where pilots talk to ATC officers via radio, due to various reasons, which will be discussed later.

- We are interested in building special **ASR apps** for ATC applications.

ERAU ASR Work

- Several faculty members and students in CoE and CoA are forming an SaLAI (Speech and Language AI) team/lab to develop tools for ASR for Radiotelephony applications.
- A general theme is to develop ASR tools and models that work best for ATC applications.
- ASR for ATC is different than that for general applications as ATC uses special aeronautical phraseology, which is different than everyday English, designed to reduce the risk of miscommunications.
  - Aeronautical phraseology is shared by pilots and ATC controllers.
  - Many new pilots have difficulties with this phraseology, and special training is needed.
- Currently, we are interested in developing an **ASR-based training app** that can be used for aeronautical phraseology training for pilots.

This app will be called RTube, a combination of Radiotelephony and YouTube type of app.

Necessity of RTube

- The best aeronautical phraseology training, especially after pilot students have gained enough aeronautical knowledge, is always real-world training.
- Listening to the audios from LiveATC does not help much for two reasons.
    - Many people just cannot decipher what the speakers say due to the special phraseology, rapid speeds, and low quality of audio signals.
    - Even if the words can be deciphered, they often do not make much sense without knowing the context of where the aircraft is.
- Our RTube will solve these two problems together.
    - It will transcribe the conversations between the pilot and ATC so that the trainee will know that was said.
    - It can display the airplanes on the map so that the trainee can easily know the flight phase and situation to make sense of the conversations.

Requirements for RTube

- High accuracy transcription.
- Airplane callsign identification.
  - Each airplane is associated with a callsign.
  - With callsign identification, we can transcribe only the communications associated with certain callsigns. The transcription can be displayed beside the airplane with correct callsign.
  - This selective display of transcripts can declutter so that the trainee can focus on the interested airplane(s).
  - Callsigns can be chosen from ADS-B data.
  - Callsign identification is not trivial, and the abbreviation of callsigns can complicate the problem.

Requirements for RTube (cont'd)

- Frequency selection.
  - A certain ATC facility has many different frequencies, each for a different set of functionalities, such as Ground for taxiing.
  - An airplane changes frequencies several times during the flight.
  - We should be able to track the changes of frequencies so that we can follow the entire procedure of departures and arrivals.
- Approaches for frequency selection.
  - We can use frequency identification performed during the transcription of speeches.
  - We can also monitor all the frequencies related to a certain ATC facility to pick up the activities of the callsign of interest.
  - A combination of the above two can help.

Requirements for RTube (cont'd)

- Display mode.
  - Live mode. Used the display the airplane positions using the real-time ADS-B data and use the LiveATC live stream. Airplane icon will change when speaks or receives. Transcripts will be displayed in real time as the audio stream comes.
  - Review mode.
    - Can display transcripts ahead of the audio according to setup.
    - Can replay audio clips.
    - Can align communication with flight path to identify where and when and on which frequency the transmission occurred.
- Other controls.
  - Can select / deselect callsigns (airplanes) of interest. Monitoring multiple airplanes is possible.
  - Common GUI functionalities, such as zooming and panning, will be supported.

Target Audiences of RTube

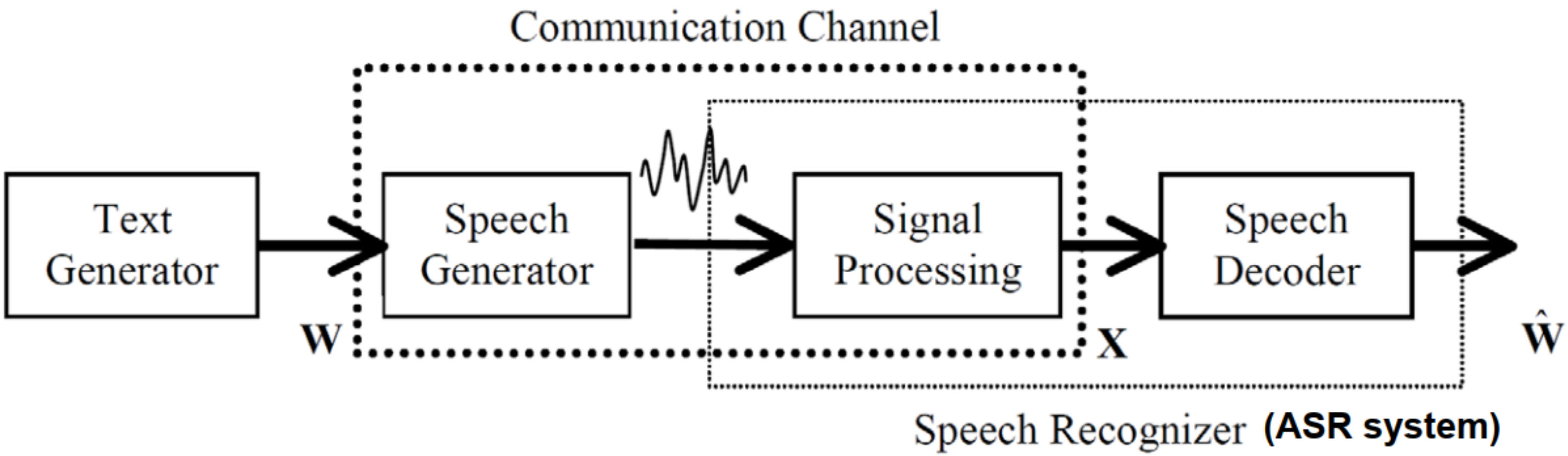
- Flight students
  - Self-paced contextualized comprehensible language training.
- Airlines
  - Prepare pilots with the local terminologies of the airports of operations, especially in safety-critical phases of flights.
- Flight instructors.
  - Debrief radio communication performance from completed flights.
- Flight safety.
  - Monitor radio communications within an operation to provide support for aircraft in needed.

Additional Possibilities in the Future

With good ASR models, we can perform many other related work, which can also be NLP-based.

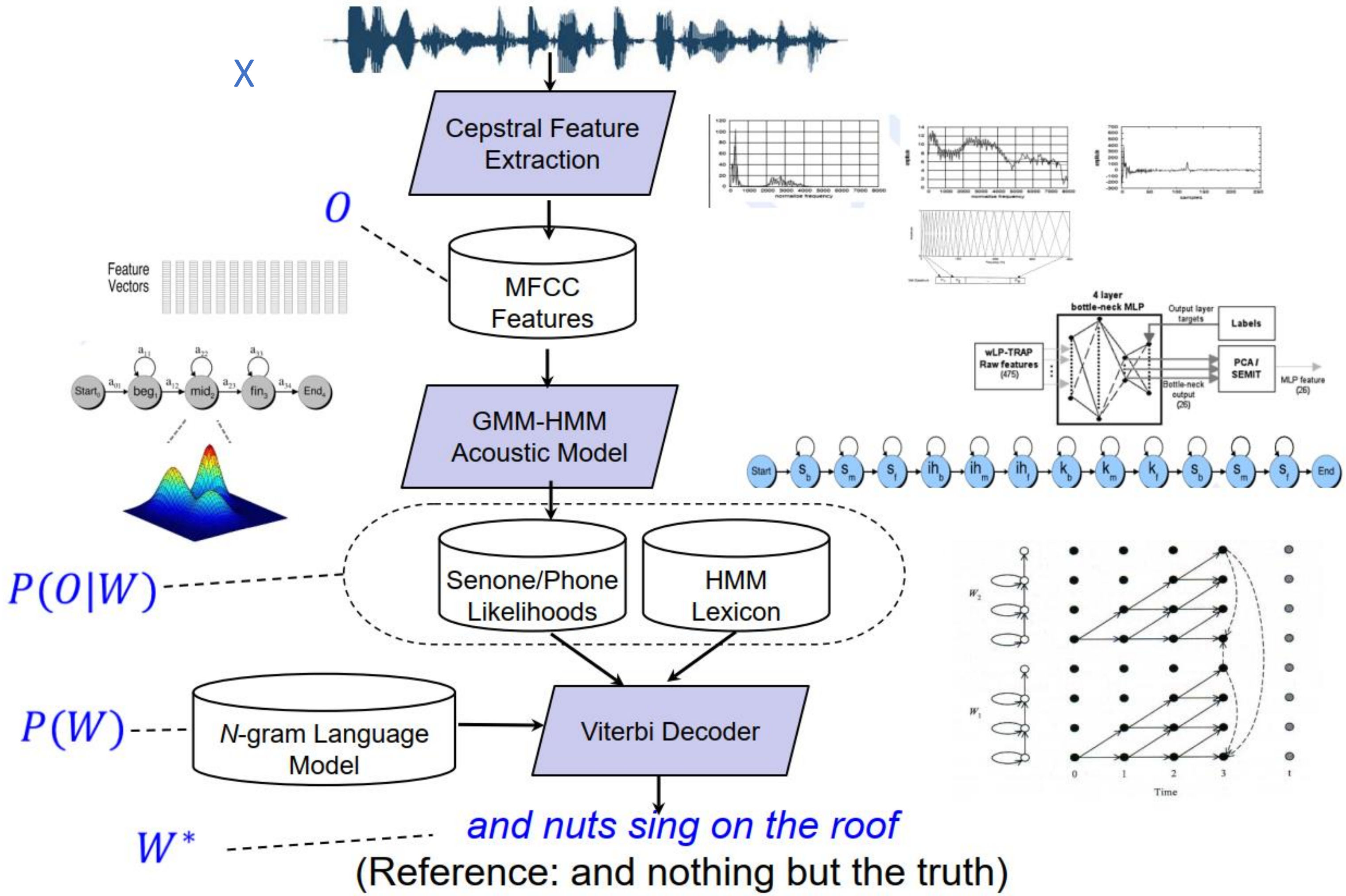
- Identify types of communication: standard phraseology; not-standard; or plain English.
- Identify miscommunications by comparing the ATC instructions and pilot readbacks.
- Monitor communication trends across fleet to identify areas of further standardization and training.

A Simple Source-Channel Model for ASR



- W is the sequence of words from a certain speaker. It is called an utterance.
- X is the speech signal. We can extract O, the feature from X. O can have different format, but mostly in **frequency** domain.
- $\widehat{W}$ is the sequence we want to obtain given X or O.

Traditional GMM-HMM-based ASR Systems



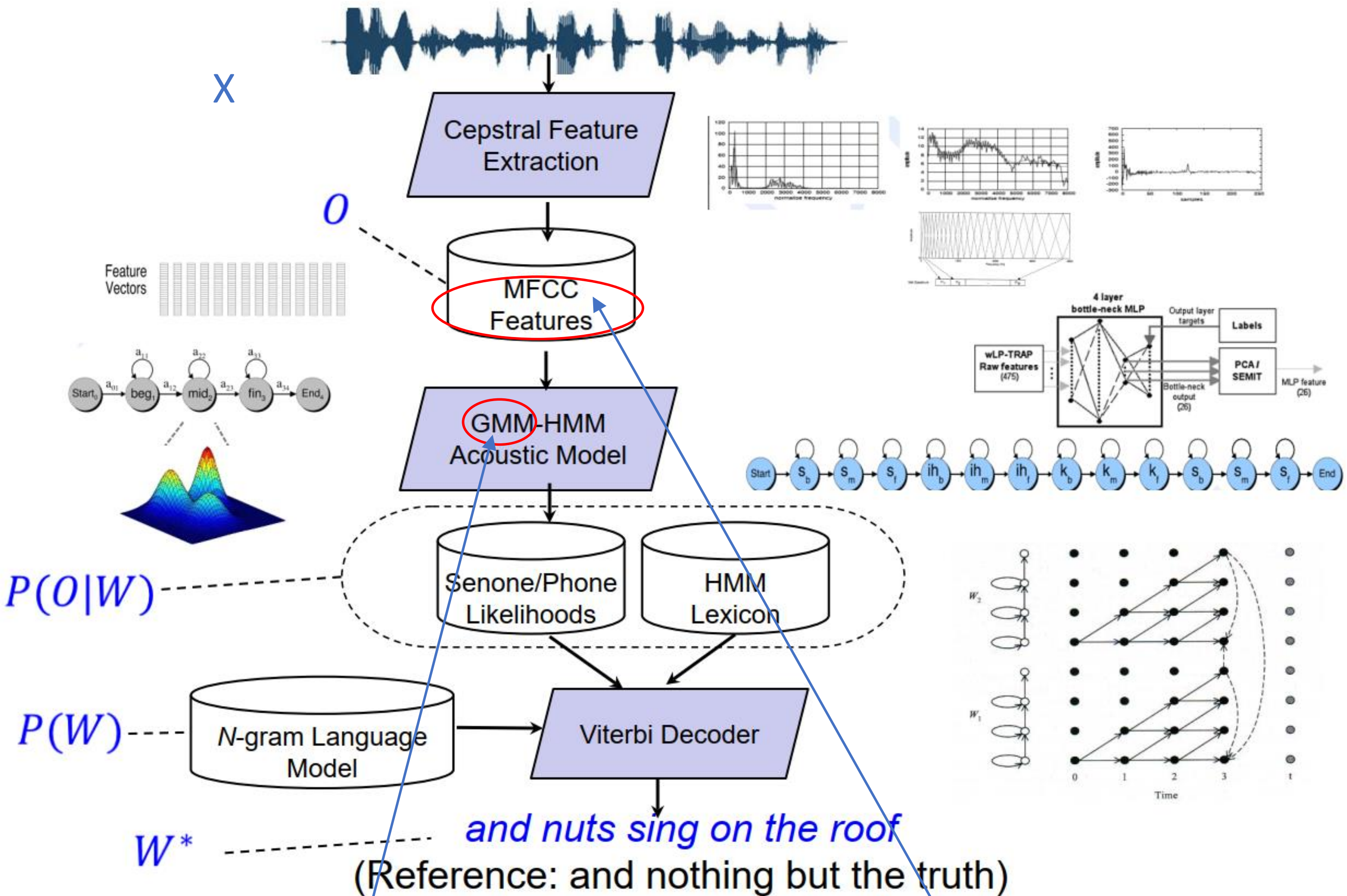
No worries, we will get you covered for each of the parts if you choose to do this project!
Some terms are explained on the next page.

Explanation of Used Terms

- Phones and senones are used to model the pronunciation of words in an utterance.
  - These phones comes from a lexicon dictionary.
  - To better model the transition of phones, we use tri-phones.
  - A state of a tri-phone is called a senone.
- HMM = Hidden Markov Model, which is used to model the transitions of tri-phones.
- O is expressed as a vector using MFCC, Mel-frequency cepstral coefficients, with minimum dependences between different dimensions.
- Given a senone, a state of the tri-phone, the distribution of O can be modeled using a simple Gaussian Mixture Model (GMM).
- Machine learning will be used to train the HMM and GMM models given speech signals and labels.
- When the models are trained, a decoder can be formed based on the Viterbi algorithm.

Again, we will help you understand each part if you choose to do this project!

Modern DNN-HMM-based ASR Systems



By replacing MFCC with a higher dimension vector called FBank and GMM with a deep neural network (DNN), we have the modern DNN-HMM-based ASR systems.

Modern End-to-End ASR Systems

- DNN-HMM-based ASR systems work better than the GMM-HMM-based ASR systems in terms of WER (word error rate) due to the better performance of DNN in modeling the features than GMM.
- There are many different forms of DNN-HMM-based ASR systems.
- Language models, which prescribe the probability of a certain word given certain other words, are important for the HMM-based ASR systems.
- A new type of ASR system put the acoustic model (HMM) and the language model together to get from O to W directly, this is called an E2E (end-to-end) system.

- DNN-HMM-based ASR systems usually work better for small dataset cases: up to 100 hours.
- E2E-based ASR systems have better potential when we have more data: 1000+ hours.

ASR Tools to Use

We have done some basic work already and have the following suggestions for the tools to use for long-term development of the project beyond senior design:

Kaldi, the DNN-HMM-based tool:

- Low-level tools are developed using C++.
- Middle-level tools are developed using Python and Perl.
- High-level tools, called recipes, are developed using Bash.
- DNN are built based on configurations, and many type of DNNs are supported.

NeMo, a collection of E2E tools:

- Low-level code are developed using PyTorch.
- High-level are based on Python notebooks.
- DNN are built based on configurations.
  - DNNs in NeMo are more configurable compared to those of Kaldi.

We would like to play with both ASR tools. We will mainly focus on the high-level coding and configurations for both tools. We will read low-level code as needed.

Two ASR Teams for the Work

Both the above tools are HUGE, and we prefer to have two teams to work in parallel:

The Kaldi team:

- 4 to 5 students with interests of learning Bash, Python, and C++ programming while doing.
- A good understanding of one or two programming languages is required.
- Will focus on finding the best performing ASR models for our ATC dataset using Kaldi.

The NeMo team:

- 4 to 5 students with interest of learning Python and PyTorch while doing.
- Machine learning or Web app development background is helpful.
- Will focus on callsign identification based on our previous ASR work.
- Will also focus on web app development.

We will get you started smoothly. Team members can switch team later if so desired and approved by course instructors.

ASR Work Performed by a Previous Senior Design Team

- Worked on ASR models based on NeMo.
- Created a web interface with a map and plane markers that showed aircraft position as close to real-time as possible. Airplane position data comes from ADS-B using API.
- Enabled live playback of ATC communications through LiveATC.
- Trained and tested several ASR models using our ATC dataset.
- Integrated a trained ASR model that transcribes live communications from LiveATC.

Improvements During the Summer

- Changed the web server from Plotly Dash to Flask.
- Added flight paths to the map.
- Added transponder waypoint markers to flight paths.
- Changed the display of aircraft information on the map.
- Added a view for airport information similar to the aircraft information view.
- Changed audio source for KDAB frequencies from LiveATC to NEAR Lab (higher fidelity).
- (WIP) ASR model runs in a separate process, so the web server isn't blocked while the model is transcribing.
- (WIP) Changed the way audio data is fetched so it works more generically, instead of specifically for LiveATC.

Screenshots for Previous Work

Airport Properties

Identifier	KDAB
Name	Daytona Beach International Airport
Position	29.179899°N -81.058098°W
Elevation	34 feet
Region Name	Florida
Municipality	Daytona Beach
GPS Code	KDAB
IATA Code	DAB
Local Code	DAB
Website	null

Tower Frequencies

132.875

0:00 /

Transcribe

118.100

0:00 /

Transcribe

118.850

0:00 /

Transcribe

120.700

0:00 /

Transcribe

121.900

0:00 /

Transcribe

125.350

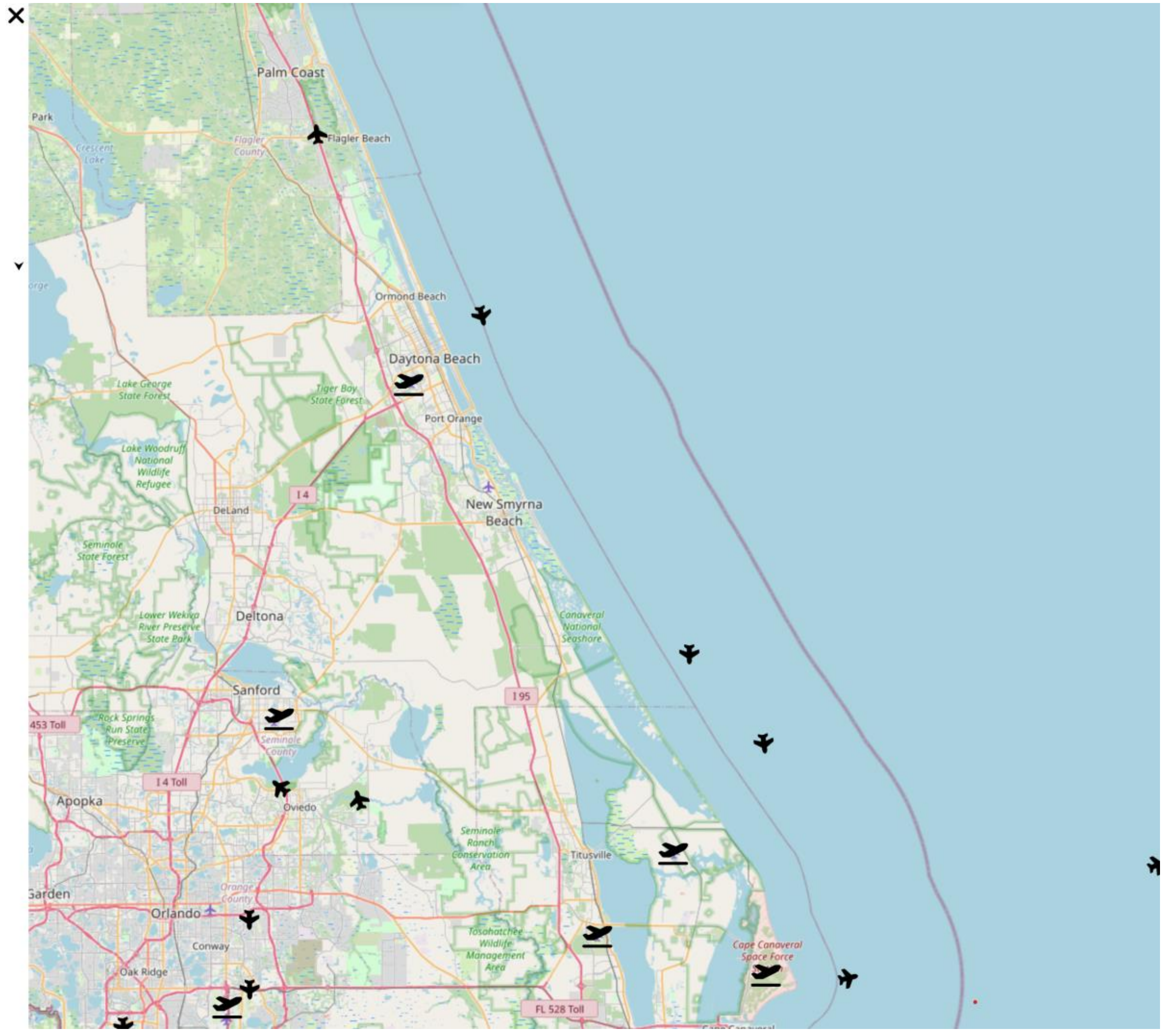
0:00 /

Transcribe

125.800

0:00 / 0:00

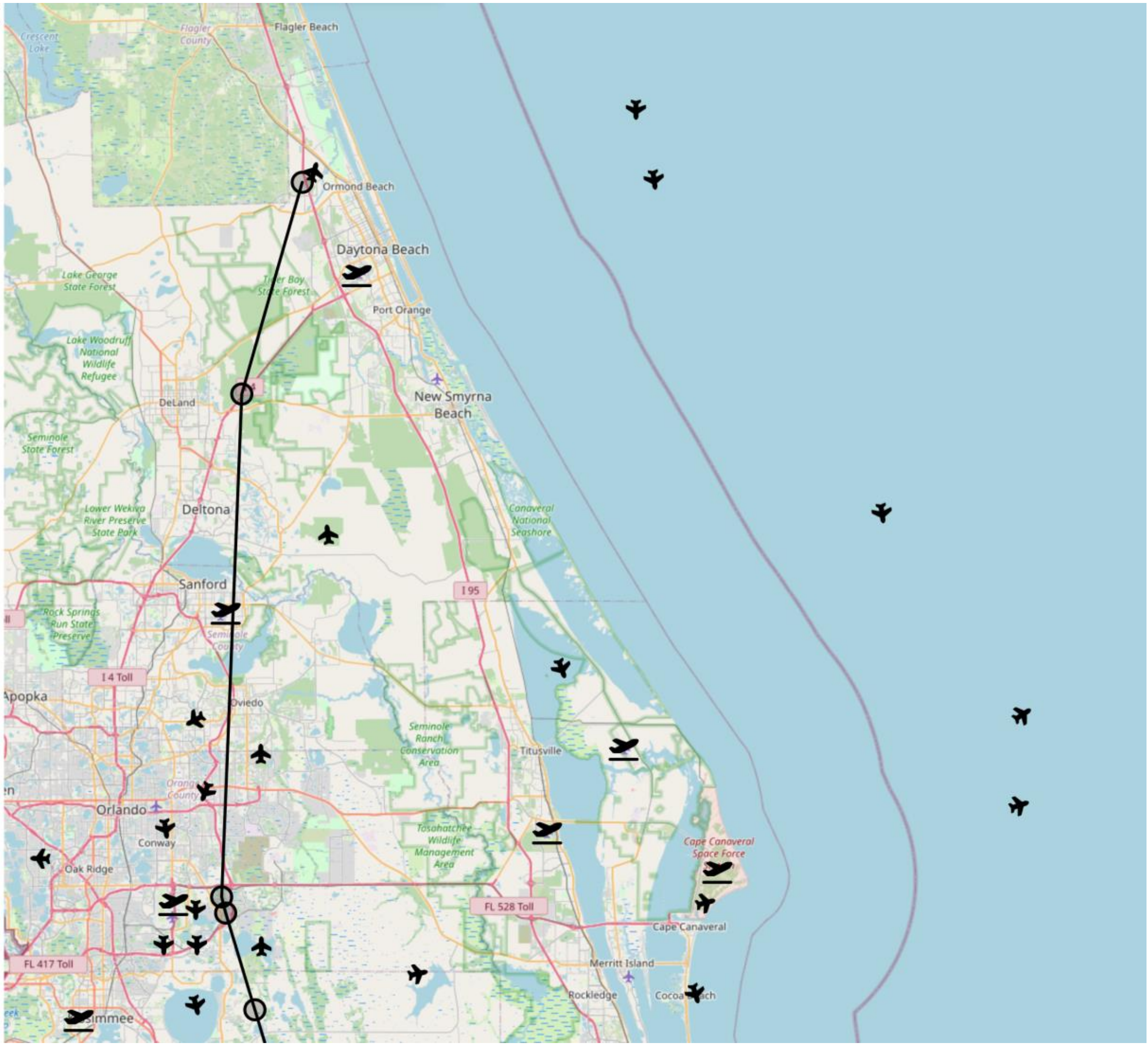
Transcribe



Screenshots for Previous Work (Cont'd)

Plane Properties

ICAO 24-bit Address	A21839
Callsign	ENY3616
Country Origin	United States
Time of Last Position Report	Tue Aug 29 2023 21:16:25 GMT-0400 (Eastern Daylight Time)
Last Contact (time)	Tue Aug 29 2023 21:16:25 GMT-0400 (Eastern Daylight Time)
Position	29.2196°N -81.156°W
Geometric Altitude	11353.8 meters
On Ground?	No
Velocity	245.42 meters per second
Heading	15.81°
Vertical Rate	0 meters per second
Squawk	3772
Position Source	ADS-B
Plane Category	Unknown



Requirements for the Kaldi Team

- One of the world-class ASR teams has developed several ASR models using Kaldi for ATC applications. Results have been published.
- We plan to duplicate their work and try to improve based on their results.
- Specifically, we will do the following:
    - Understand the current ASR models.
    - Adjust the ASR models using our 30-hour ATC dataset.
    - Compare the performance of the trained models.
    - Experiment with other popular models.
    - Apply the best trained models in an online speech recognizer, which transcribes the live stream.
        - The online speech recognizer needs to be able to run as a process so that it can be called by RTube in the future.
    - Experiment with callsign and frequency identification.

Requirements for the NeMo Team

- ASR:
  - Callsign and frequency identification from speech signals.
  - ASR models with improved WER performance.
- RTube integration:
  - Program the functionalities of RTube listed in previous slides.
  - Integrate ASR transcription and callsign / frequency identification with NeMo-based models.
  - Experiment with the ASR transcription and callsign / frequency identification with Kaldi-based models.
- Additional functionalities of RTube:
  - Add a database to keep track of calls between pilots and controllers (audio, transcript, time, date, location, callsign).
  - Add different icons for different aircraft categories and types.
  - Allow “breadcrumbs” to update along with icon movement.
  - Configurations for different, location-specific, use cases, i.e., for use at different flight schools.

Questions?