Radar-Cued Camera Data Collection System

STUDENTS: AKHIL MANDALA, PAUL CHUNG, SIDHARTH DAGA, CHRISTIAN BLEVENS

Problem Statement

- Echodyne is seeking a large dataset of labeled images to train their ML system
- Our goal was to create a field-deployable system to capture images of objects tracked by a radar
- Echodyne provided a radar, which was used to detect objects, and provided a camera, which would point to the detected objects
- Our system's goal was to have a user select a radar track from a UI, a camera would then point to the selected track, and the user would then be able to capture images and store label data of the selected track into a database
- Our team's goal was to create a solution that worked with Echodyne's existing codebase

Requirements

The following are high-level requirements that our system needed to abide by:

- System must be safely and easily field-deployable – non-technical to operate
- System must display tracks on a map display in real-time
- System must point PTZ camera at selected radar track and display video feed
- Users must be able to manipulate orientation of the camera through the UI
- Users must be able to capture images
- Images and associated data (track ID, lifetime, label, time captured, etc.) must be saved in a database

Figure 1: The system's setup at Gas Works Park.

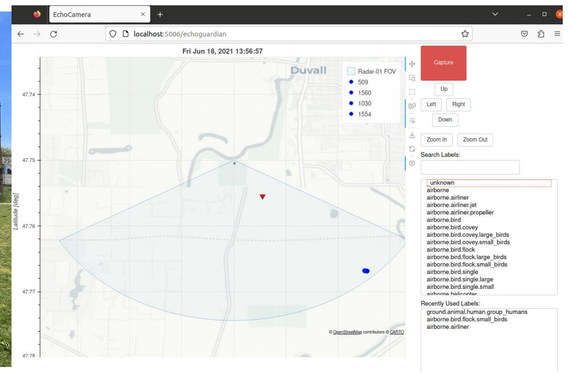

Figure 2: As shown above, EchoCamera is used to select tracks, manually manipulate camera movements, add labels, and capture images.

Implementation

EchoCamera:

- The UI, which we refer to as EchoCamera, is created from a Python library called Bokeh and allows a user to manipulate camera movement, view tracks, add labels to tracks, and capture images
- A track is selected from the UI, and a camera feed displays the selected track using OpenCV

Camera Steering:

- Camera steering can be done with absolute coordinates or relative movements
- Absolute coordinates are used when a radar track is selected and the real-world location data is processed to give coordinates that have the camera facing it
- Absolute coordinates will continuously be calculated while a radar track is selected to accurately follow a radar tracks movements
- Relative movements can be used via the buttons in EchoCamera

Database:

- Images taken from field usage are stored locally
- Image metadata is stored as a JSON, and uploaded to a permanent PostGres database for later use in the labeling tool
- Image metadata includes all information necessary to retrieve associated track radar data, and any live-annotated labels
- A permanent database hosts all label data, including edit history

Labeling Tool:

- Populates an explore page in which images from tracks are shown with their respective metadata from querying the database
- Metadata shown are the fields that can be filtered upon, including radar SN, camera SN, label, labeler, live-labeled, location, and date
- Labeling page enables users to apply label to image using similar drop-down functionality as live-labeling tool (Figure 2)

Discussion/Future Work

Current System:

- Once the correct offset has been applied to the initial camera azimuth, the system is able to display the feed of tracks selected from the EchoCamera UI
- Pictures can be captured from the video feed and saved to a permanent database
- Tracks can either be live labeled through the EchoCamera UI or through the labeling tool over the permanent database

Future Work:

- Automatic track selection based on track confidence level
- Automatic image capture based on target pixel strength in feed view
- Machine learning model to label images of tracks
- Recurrent Neural Network to label tracks solely based on their radar data using database of tracks to labels generated by our system

Conclusion/Acknowledgements

- We used simulations written in Unity and extensive testing to verify the coordinate transformation math beforehand. Extensive testing can reduce workload ahead of time and helps isolate causes of error.
- We wrote extensive documentation for system setup, which helped accelerate live field testing and debugging.
- Throughout the class, our team collaborated closely with Echodyne to determine project requirements and get assistance with setup and development. We appreciate their help immensely.

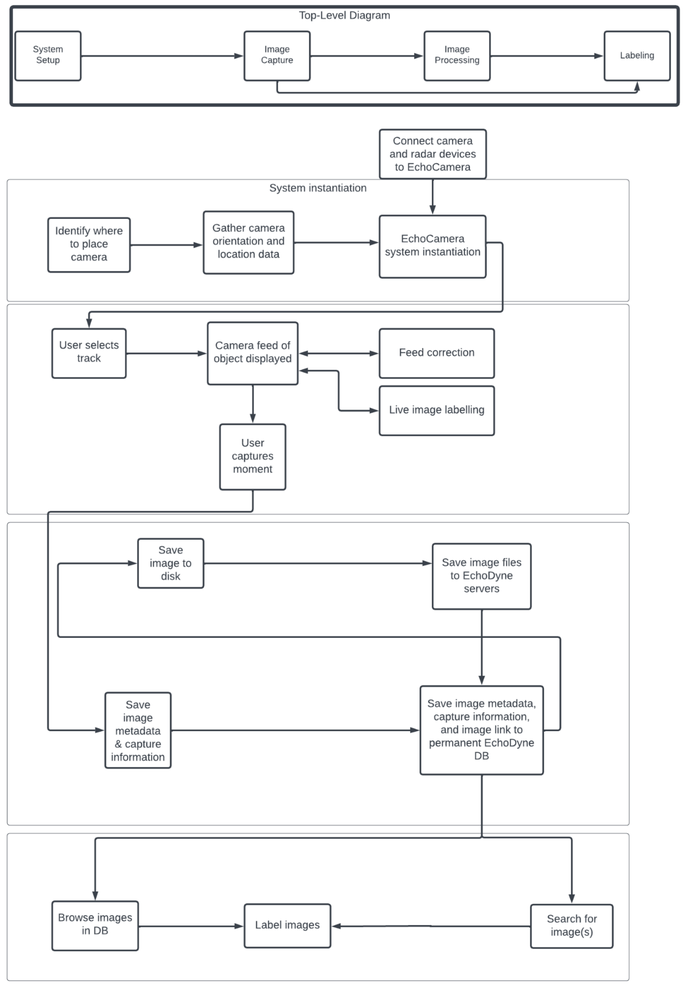

Figure 3: Flow diagram of our system's implementation.

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

ADVISERS: JACK HUNT, MINGFEI CHEN

SPONSOR: Echodyne