College of Engineering Computer Science

Kinect and Radar Data Visualizer (KaRD)

Joseph Green, Dr. Chris Crawford, Dr. Sevgi Gurbuz, Dr. Evguenia Malaia
Auburn University

AUBURN UNIVERSITY

Introduction

Radar tech has many applications in the world, from guiding airplanes to being the reason you get a speeding ticket. And there are more applications to come. Dr. Sevgi Gurbuz, my P.I. is working on a way to use radar to detect sign language. Her plan is to use two properties of radar: Micro-Doppler Effect and Range, to do this:

- Micro-doppler: see how fast an object is moving (more precise than doppler)
- Range: detect the distance between an object and the sensor

The most common way to detect sign language is using video data, specifically the Microsoft Kinect. Radar would be better because

- Better at motion detection, because of the Doppler Effect
- Less invasive of privacy, only uses waves and not video

Dr. Gurbuz ran an experiment to compare to the two and needed an app to sync the data, which is where this project comes in. The goal of this project is to create an app that can compare video data from the Kinect to radar data from the sensors with the two being synced to make the comparison easier.

Methodology

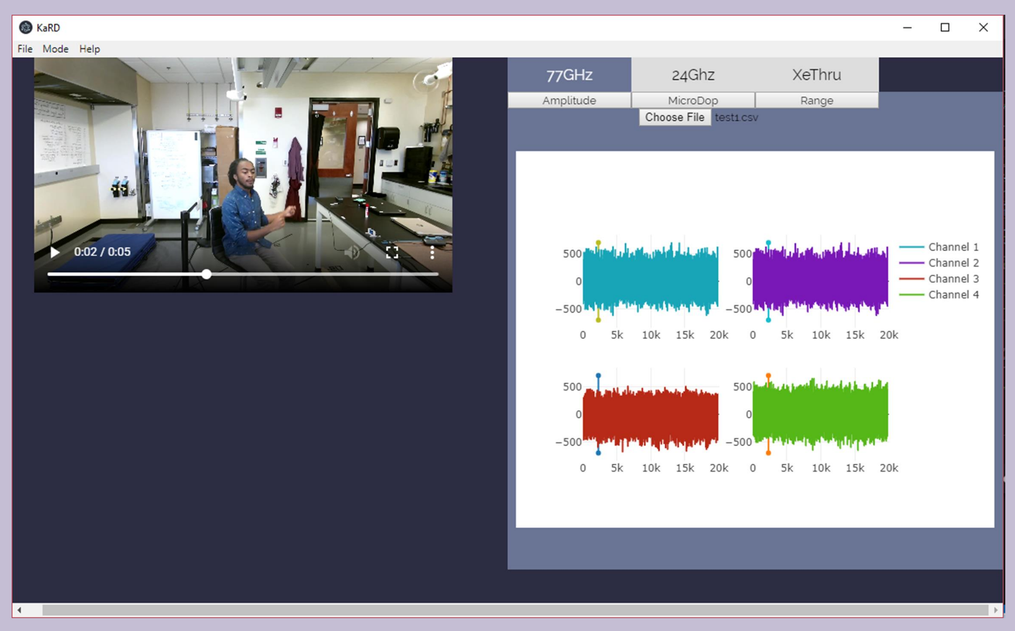
Pre-App:

- MATLAB was used to the generate images for the Micro-doppler and Range graph for speed's sake.
- Isaac Wang's XEFExtract was used to convert the video file gotten from the Kinect into a readable format for HTML

App-making:

- Coding done in JavaScript and using the Electron for app development and Plotly for rendering the graphs
- How it works in English:
  - When a tab (77 GHz, 24GHz, XeThru)/ button mode (Amplitude, MicroDop, Range)/Choose File is clicked, all the graphs are purged and the app goes through the selected source file to process and graph the data in the Plotly in the current tab.
  - If the button mode is MicroDop or Range, instead of processing the data, the app uses a preprocessed image of the data to instead for speed.
  - As the video plays, it tells the app the current time, and the app uses that information to slide the bar across the graphs

Design



- [Above] Here's the KaRD app in the 77GHz-Amplitude mode, with a video of me swing my arms

- The app is divided into two sections: one for video data and section for radar data.
  - Tabs for the different radar models, in this case 77Ghz, 24Hhz and XeThru
  - Buttons for the different formats of the data gotten from it (framing): Amplitude vs Times, Range v. Time, and Micro Doppler v. Time
- What's being graphed is controlled by the Choose File button
  - With the Amplitude v. Time, the data is graphed directly on the, for the others, an image is uploaded and used.
- A vertical bar is drawn over the data and moves along with the video, showing which spikes in Amplitude/doppler effect/range correspond to what action in the video.

Conclusion

- Fulfills the goal, although not as optimally as I would've hoped. Ideally all the data would've been processed within the app
  - While regular radar data (Amplitude vs Time) isn't hard to graph, the other types of data that give more information aren't as user/JavaScript friendly
    - The Micro Doppler and Range vs Time datasets proved too much for Plotly to render effectively
      - As a side step I used an image received from processing the data in MATLAB and made it the background of image of an essentially blank graph

Future Work

- Find an alternative to Plotly that is able to render results of the Micro-Doppler and Range within the app
- Integrate the XEFExtract tool into the JavaScript to allow the app to convert the Kinect files instead of having to do the conversion outside
- Work the other Kinect data streams to video for a more anonymous video to allow for
  - Create tabs for the video to alternate between different types of data
- Clean up cosmetics

Acknowledgements

[1] "CSV-Reader" [Online]. Available: https://www.npmjs.com/package/csv-reader. [Accessed: 20-June-2019].
[2] "Electron" [Online]. Available: https://electronjs.org. [Accessed: 20-May-2019].
[3] "Plotly.js" [Online]. Available: https://plot.ly/javascript. [Accessed: 12-June-2019].
[4] Wang, Isaac "XEFExtract." [Online]. Available: https://github.com/Isaac-W/KinectXEFTools. [Accessed: 04-June-2019].