Flying Under the Radar

Team A6
Angie Bu, Ayesha Gupta, Linsey Szabo

User Requirements

Universal Drone Attachment

Maximize usability and cover large areas despite difficult terrain

Fire SAR Missions

Reach areas where traditionally used infrared fails

Break Cost Barrier

Our mmWave radar application is cheaper than drones currently used for SAR

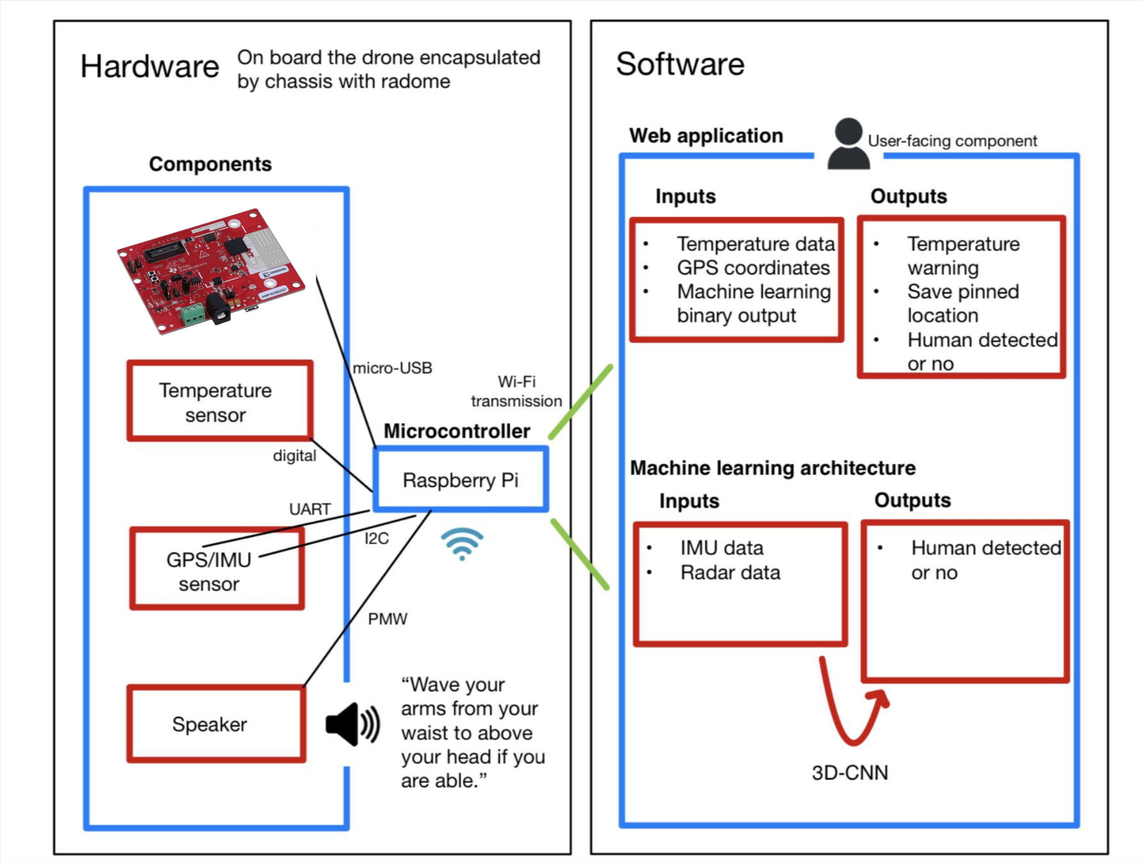
Human Detection

Detect and save locations of victims for efficient rescue

Solution Approach

Design Changes

- Hardware
  - Radar changes
- Machine Learning
  - Large data difficulties
  - PyTorch -> Tensorflow
- Web app
  - HERE Maps API



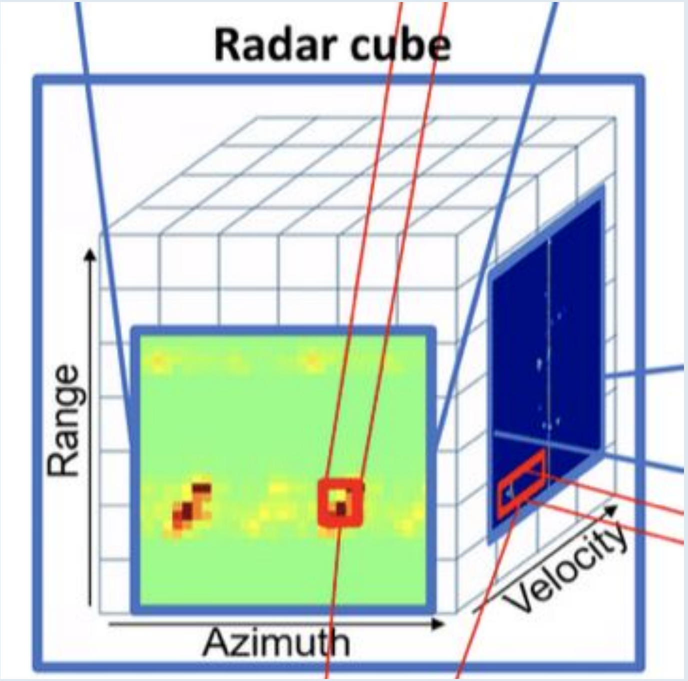
Complete Solution

- Radar capturing live scenes of moving people
  - Interactive with obstructions
- Detection output
- Web application with map and GPS data on other monitor
- Immediate display of sensor information

Data Collection

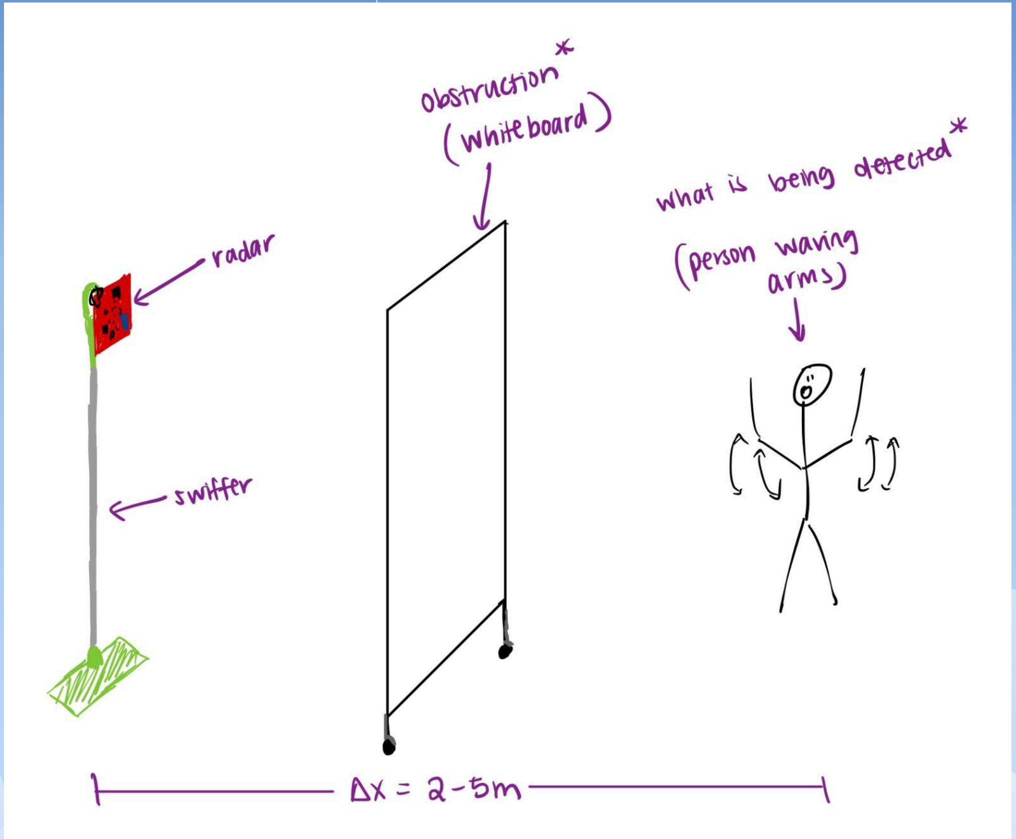
Palffy, Andras, et al. IEEE ROBOTICS AND AUTOMATION LETTERS, 2020, *CNN Based Road User Detection Using the 3D Radar Cube*, https://arxiv.org/pdf/2004.12165.pdf.

- Neural network trained on our dataset of 3600 samples
  - 1800 samples with humans
  - 1800 samples without humans



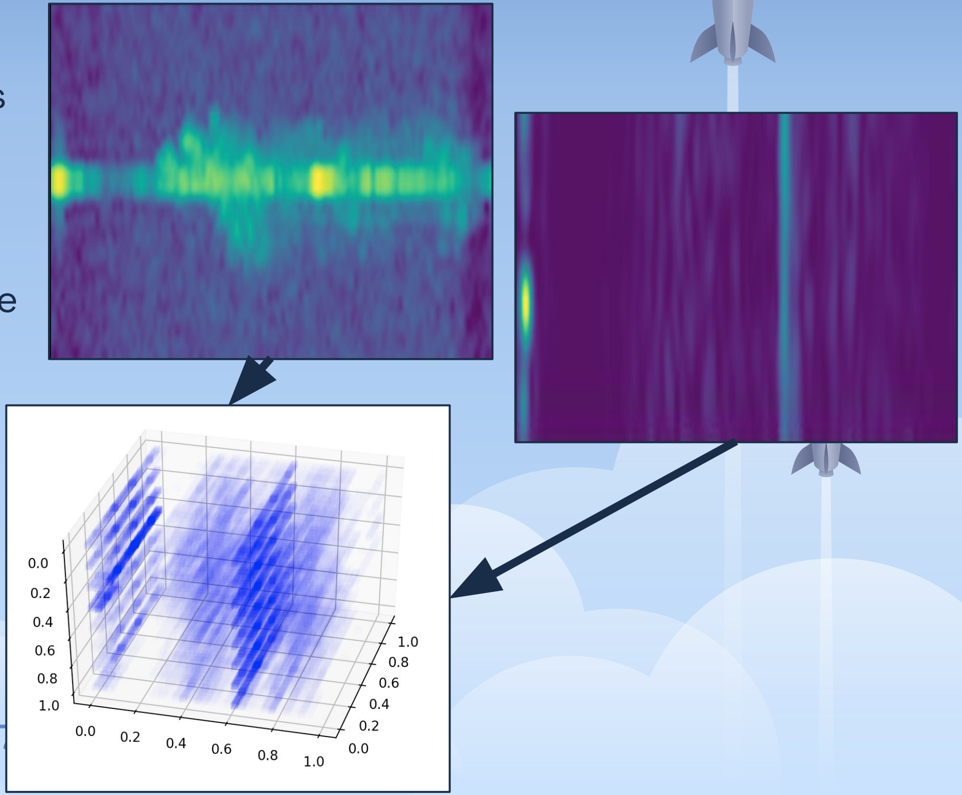
Data Collection

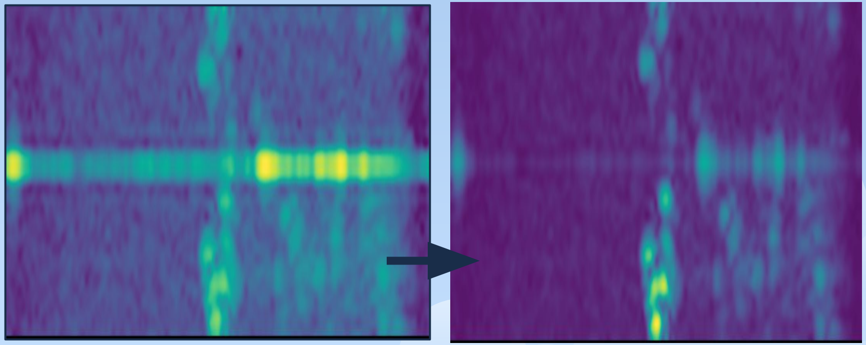
- Attach radar to Swiffer
- Capture various scenes
  - Obstructions
  - Moving humans
  - Breathing humans
  - Moving objects
  - Static
- Distances of 5 m and less



Test, Verification and Validation

- **Preprocessing pipeline**
  - Get range-doppler & range-azimuth maps
  - Project both maps into 3D range-doppler-azimuth space
  - Denoising to emphasize doppler signature typical of a human target



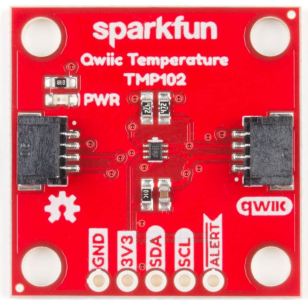


Test, Verification and Validation

Hardware

- Temperature sensor
  - Tested ambient temperature with Arduino
  - Can validate with ambient thermometer
- Speaker
  - Needed new speaker for use case

Have



TMP102

Quieter Speakers

Need

Ambient Thermometer

Louder Speakers

Test, Verification and Validation

ML

1 PyTorch + Gent University Dataset

Training examples	175
Validation examples	75
Epochs	15
Validation accuracy	45%

2 TensorFlow + Our Own Training Data

Training examples	2516
Validation examples	1079
Epochs	35
Validation accuracy	99.63%
F1 Score	1.00

3 Testing Metrics

Testing examples	700
Testing accuracy	58.51%
F1 Score	.50

Next

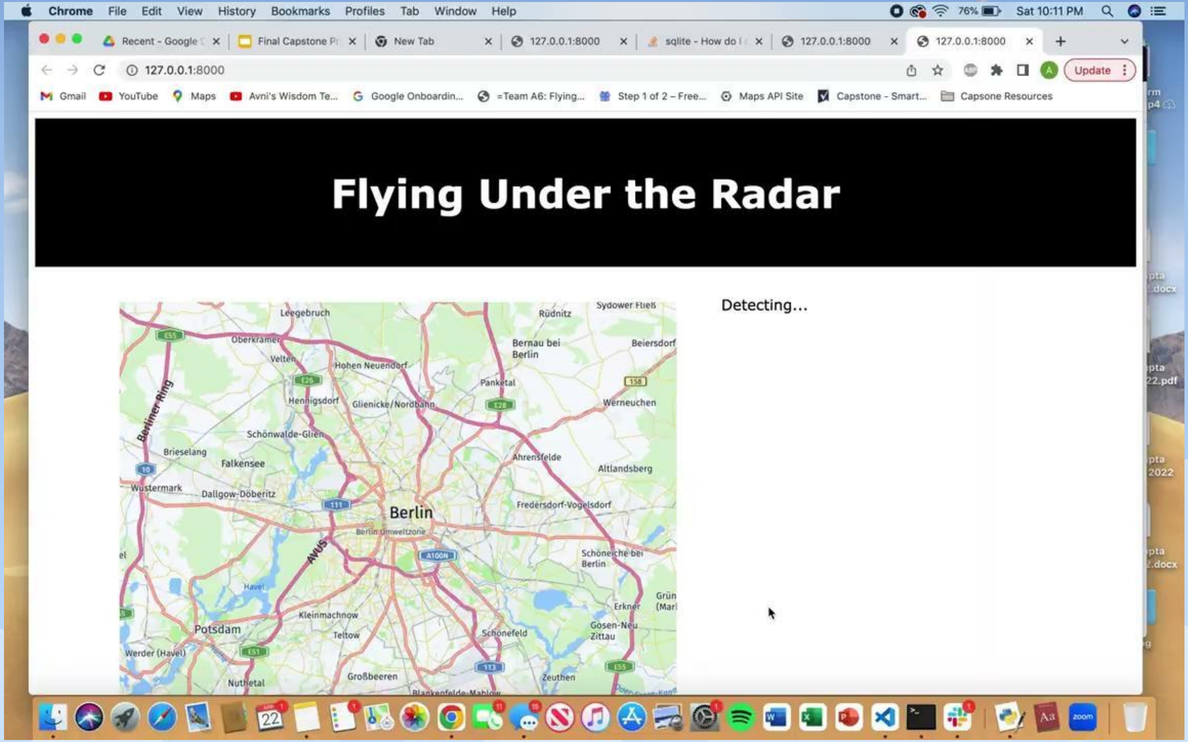
- Training on more data to increase F1 score
- Real time: 173 ms inference time

Design Requirement: F1 score = 0.7

Test, Verification and Validation

Front end

- ➢ Sending data to web app
  - ▻ Http requests using Python requests
- ➢ 300 ms within request being sent for location to update



Flying Under the Radar

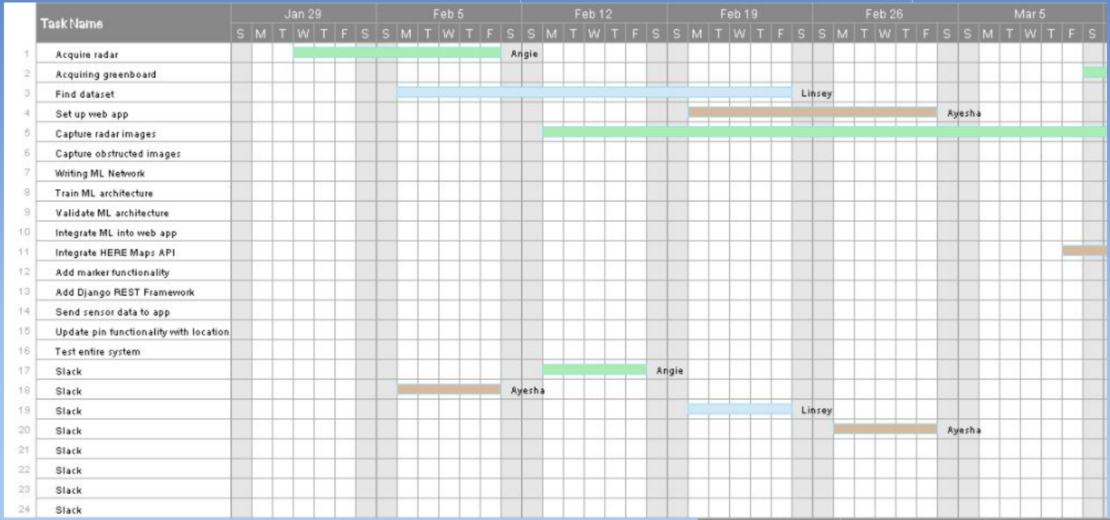
Detecting...

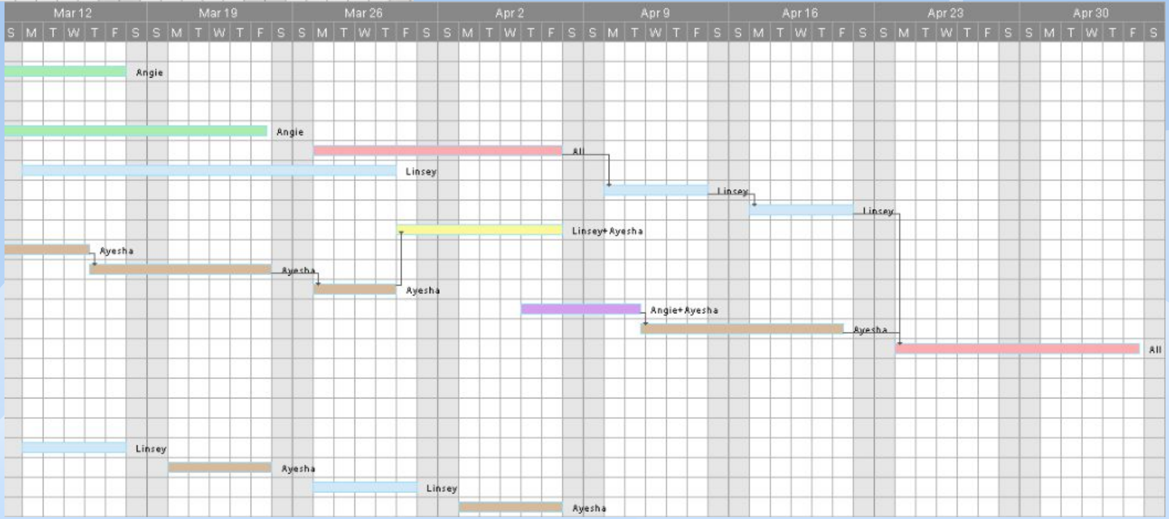
Design Metrics

Requirement	Metric
mmWave radar detection range	< 5 m
GPS localization accuracy	0.5 m
Temperature warning point	100 ° C
F1 score	0.7
Web application latency	100 ms
System latency	3 s

Changes/Delays

- ➢ Switching radars
- ➢ Switching Map APIs
- ➢ Data collection for training





	Task Name		
1	Acquire radar	Angie	
2	Acquiring greenboard		Angie
3	Find dataset	Linsey	
4	Set up web app	Ayesha	
5	Capture radar images		Angie
6	Capture obstructed images		All
7	Writing ML Network		Linsey
8	Train ML architecture		Linsey
9	Validate ML architecture		Linsey
10	Integrate ML into web app		Linsey+Ayesha
11	Integrate HERE Maps API		Ayesha
12	Add marker functionality		Ayesha
13	Add Django REST Framework		Ayesha
14	Send sensor data to app		Angie+Ayesha
15	Update pin functionality with location		Ayesha
16	Test entire system		All
17	Slack	Angie	
18	Slack	Ayesha	
19	Slack	Linsey	
20	Slack	Ayesha	
21	Slack		Linsey
22	Slack		Ayesha
23	Slack		Linsey
24	Slack		Ayesha