BLINKAH Final Test Report (Team 02)

Parker Van Roy (parkervr@bu.edu)
Raghurama Bukkarayasamudram (rbukka@bu.edu)
Ritam Das (ritamdas@bu.edu)
William Pine (wpine@bu.edu)
Ayush Upneja (upneja@bu.edu)

April 2021

Contents

1 Required Materials

1.1 Hardware

- Jetson Nano 4G
- CSI Camera
- 7" HDMI LCD Touchscreen
- USB Speaker
- Hologram 4G Modem & SIM

1.2 Head Unit Software

- OpenCV ALPR
- OpenCV MobileNet V2
- OpenCV Proprietary Lane Detection
- Erratic Driver Detection
- QtPy5 User Interface
- IBM Watson TTS & Audio Notifications

1.3 BCF

- Nodejs
- Django
- Firebase
- 2GB Shared Intel SSD Digital Ocean Droplet

1.4 Other

- Car: 1998 Volvo V70 Cross-Country Station-wagon (AKA BLINKAH Mobile Transporter)

2 Introduction to Project

What is “BLINKAH?” BLINKAH Head Units are edge-computing dash cameras that leverage computer vision to analyze and report on other vehicles on the roads.Connected to the BLINKAH Computational Frontend, we analyze data from multiple head units and keep track of vehicle behavior, allowing us to create reports of erratic drivers for insurance purposes and to send warnings to drivers in the vicinity. BLINKAH ultimately eliminates irrelevant entities in traffic stops and reinforces safe driving habits.

3 Test Plan

In this project, we plan to breakdown the test into two portions: the BLINKAH Head Unit and the BLINKAH Computational Frontend. The BLINKAH head unit involves the jetson nano, its required hardware components, interaction with the cloud, and is expected to perform the following:

- Stream road footage
  - Confirm stream functions locally
- Computer vision subcomponents
  - ALPR test to accurately read license plates from video
  - MobileNet V2 to correctly identify other vehicles
  - Confirm accurate lane detection
- Detect Erratic Driving
  - Set and track vehicle bounds
  - Simulate Vehicle position
  - Detect & report swerving
- Head Unit Frontend
  - Test vocal notifications
  - Show CV Output
  - Ensure working 4G Communication

In addition to these elements, the BLINKAH head unit is also responsible for sending the stream footage and incident report to the backend.

The BLINKAH Computational Frontend is a dashboard that polls backend data for informative visualization purposes. It is expected to perform the following:

- Poll Chronological Backend Data
  - Reported in tabular format
  - Visualize incident locations on heatmap
  - License Plate Lookup
- Administrative Dashboard

4 Set-up

4.1 Server-side

We will be hosting the Django server that needs to enable ssh connectivity in order for the head unit to communicate with the backend.

- Create a virtual environment
- Run python manage.py runserver @ip_address:PORT

4.2 Head-unit side:

We will be mounting the jetson nano and its components to the dash of the BLINKAH mobile, making sure that the camera has an optimal viewing angle for roads and vehicles. Power the Jetson and run the UI application.

5 Testing Procedures

To properly test our project, we plan to drive around select neighborhoods in Boston to gain multiple road perspectives, road conditions, and a variety of vehicles. We will test our computer vision and erratic driver detection algorithm on these vehicles and roads to generate real time reports. We expect to see all reports listed in the Django reports tab, and while all of them may not qualify as erratic driving, we expect at least of our reports that can be classified as erratic driving incidents. These reports are polled and received by the BLINKAH Computational Frontend where it should display the report along with the incident locations visualized on a heatmap.

6 Scoring Criteria

A successful final test should pass the following requirements and succeed in basic operation.

6.1 Head Unit:

- Properly stream road footage send to server
- Perform the following CV components properly
  - Collect license plates from ALPR
  - Detect vehicles from MobileNet V2
  - Detect lanes from OpenCV
- Determine if vehicle qualifies as an erratic driver
  - Report incident to BLINKAH Cloud
  - Confidence value consistently $> \frac{2}{3}$
- Frontend
  - Ensure functional UI
  - Audio notifications of incident(s) functional

6.2 BCF

- Server is working and polling incident data properly
  - Display chronological incident report in tabular format
  - Visualize incident locations on heatmap
  - License plate lookup
- Functional administrative dashboard

7 Scoring Chart

Task	Category	Comments	Result %
Stream Road Footage & Send to Server	Head Unit	- streamed at 3 fps - stored "images" in server properly	100%
Collect License Plates (ALPR)	Head Unit	- no lens for camera - ALPR didn't work due to quality	67%
Vehicle Detection (MobileNet V2)	Head Unit	- vehicle bounds mostly accurate	100%
Lane Detection (OpenCV)	Head Unit	- lane bounds mostly correct - Issue with curved lanes and turning vehicles	75%
Erratic Driver Algorithm	Head Unit	- successful erratic driver detection - worked on pseudo and real samples	100%
Report Erratic Driver Incident to Backend	Head Unit	- Successfully made report once erratic driver detected	100%
Ensure Consistent Confidence Value > ⅔	Head Unit	- Tested on 1 instance of erratic driving - Generated 1 valid report to server	75%
Functional Head Unit UI	Head Unit	- QtPy5 shows working CV, notifications, erratic driver image - Prof Osama: UI too flashy; should be simple	100%
Audio notifications	Head Unit	- IBM tts generated notification - Incident message with license plate displayed on head unit + audio	100%
Server running properly	BCF	- Hosting was successful	100%
Server polling incident data properly	BCF	- Get request successful for frontend	100%
Display chronological incident report(s) from database	BCF	- Displayed values in a dynamic javascript table object	100%
Incident Heatmap Visualization	BCF	- Displayed heatmap of incidents in Boston (geolocation data) (Static Data)	66%
License Plate Lookup	BCF	- Demonstrated success with multiple License Plate Lookups	100%
Functional Admin Dash	BCF	- Secure frontend dashboard with relevant information from DB - Easy to access features on single page with streamlined complexity	100%
Result %	→	→	92%

8 Conclusion

Given our scoring criteria, our final test was a success. We divided up our project into a variety of smaller tasks which were initially working perfectly on their own; however, when combining and containerizing the tasks for the head unit, we unfortunately faced many difficulties. Most of these difficulties were overcome, however, some of the issues could not be addressed within the given time frame.

Our initial plan was to test on multiple roads and road conditions, but our Jetson required more power than what was available to us with a portable battery or automobile auxiliary power, so we were forced to test our unit on a limited sample of roads using an extension cord to power the Jetson. We tested our BLINKAH unit using two cars, one hosting the head unit while the other simulated "erratic driving" on a two way road. The CSI camera quality was good enough to output a proper number of frames on which the CV was able to determine the vehicle and whether it was driving erratically. For testing purposes, we had the second car drive excessively "erratically" by bouncing in and out of the lane (when there was no traffic). This generated an erratic driver report which successfully sent to the backend. Due to our limited tests, we could not properly test how erratic a driver needs to be for the CV to deem them as an erratic driver and send the information to the server.

Our test was successful despite the unexpected technical issues and compatibility issues we encountered. Our CV components were working, but we could not properly test ALPR since the second car that was simulating erratic driving had a dashcam above the license plate which cast a strong shadow on the license plate that blocked some of the actual letters. However, this is not a major issue since we tested ALPR previously on other car images that were successful. BLINKAH successfully sent images of the erratic driver to the server. From the server, this information was showcased in the BCF as an incident report, and the head unit successfully played and displayed notifications of erratic drivers in the vicinity.

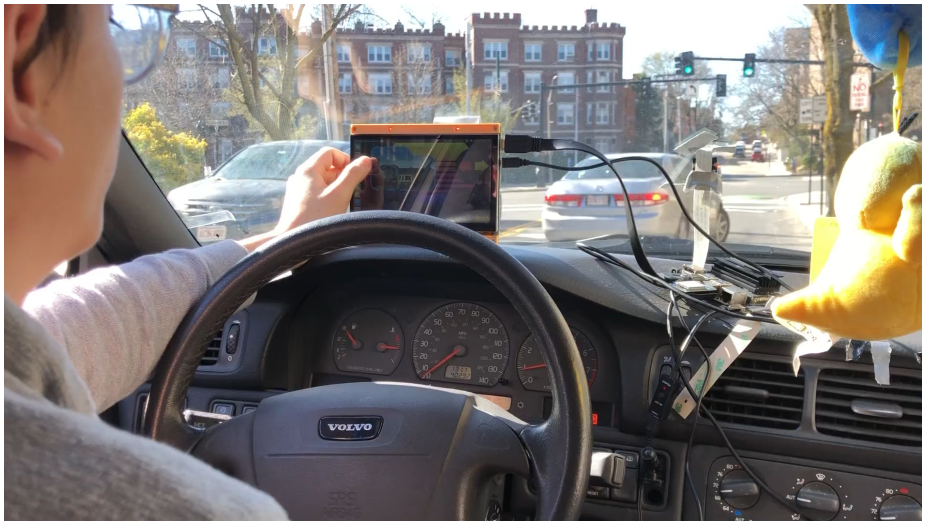
It's our plan to provide an electrical hookup inline to the user vehicle standard to dashcams during client installation. We also look to address our main technical issues and set up a second head unit for live service.

We plan to polish the head unit mounting and qt5 application during client installation as well as optimize ML. The 3D printed mount is in progress and some components were displayed during the demo, however, it remains incomplete. The qt5 application is functional but requires cosmetic cleanup and image stream for final client installation. Optimizing ML will involve optimizing buffers and threaded service startup.

9 Appendix

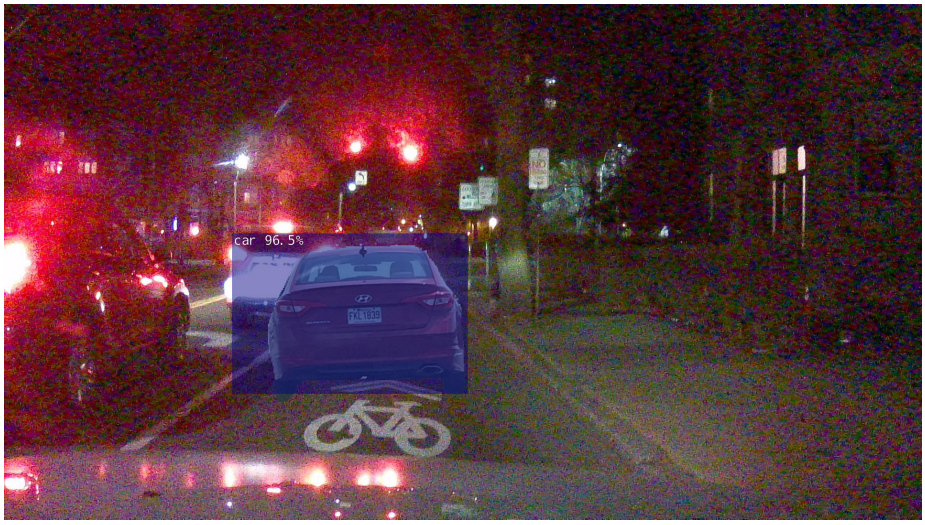
Here are some captures from the Final Test.

9.1 Head Unit Setup



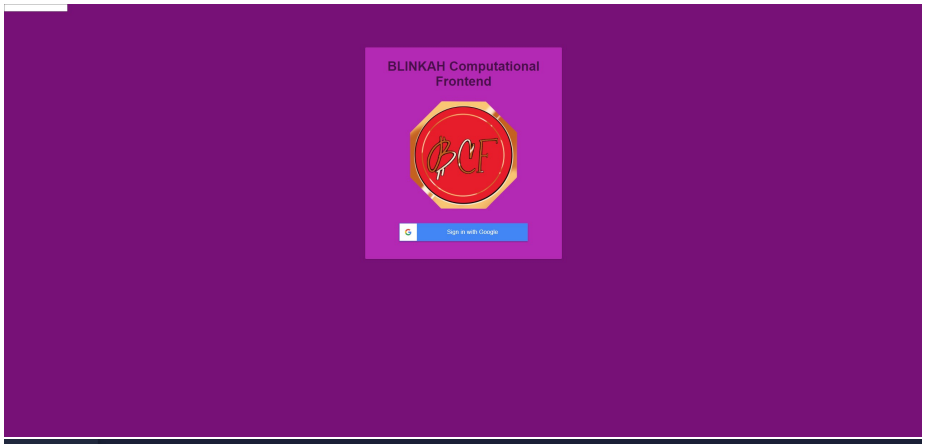
9.2 Erratic Report Photograph

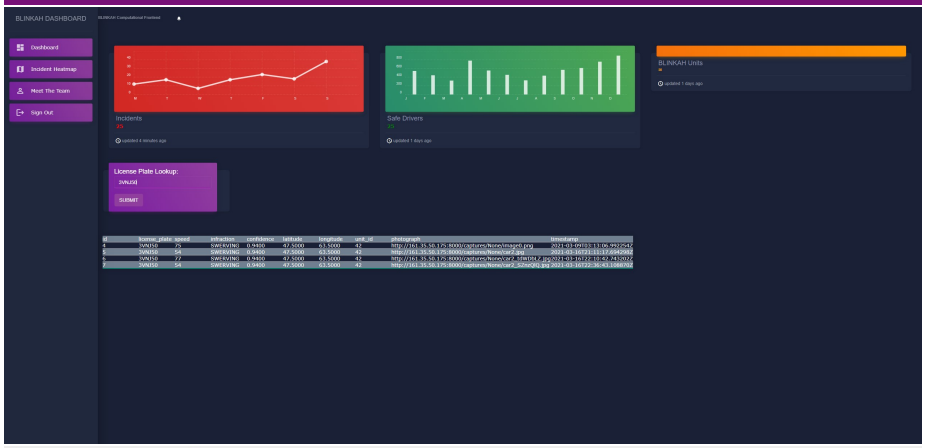
9.3 Pseudoreal Testing

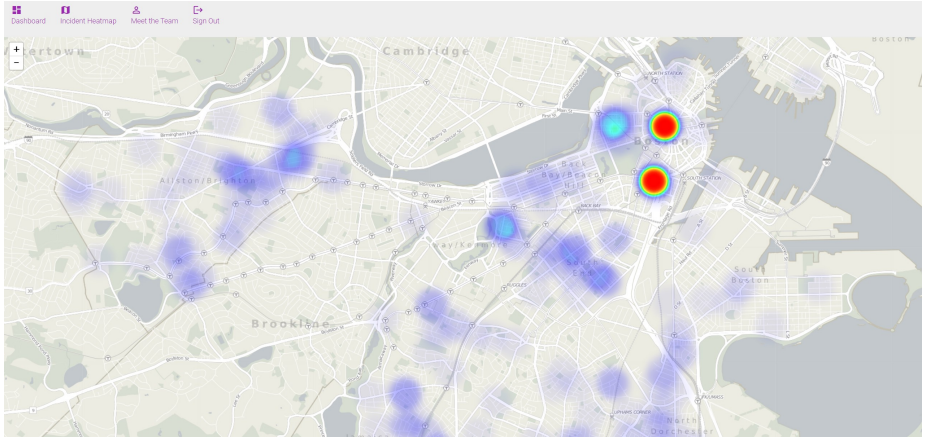


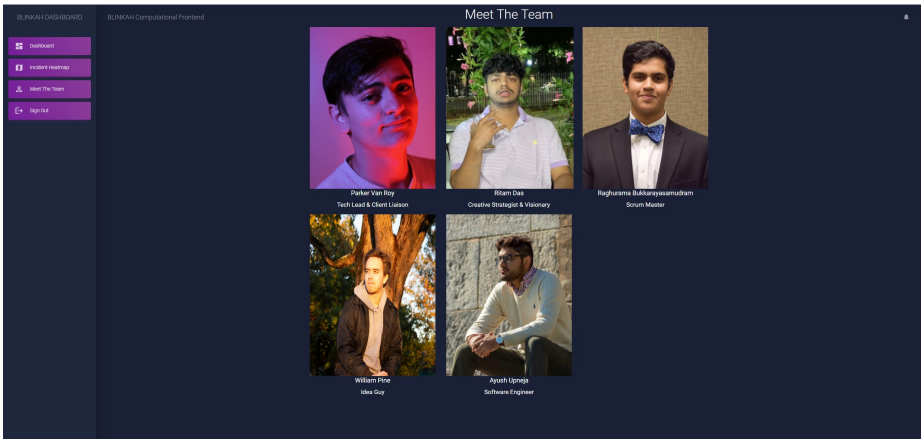
9.4 QtPy5 Application Demo

9.5 BCF


BLINKAH Computational Frontend
BCF
Sign in with Google


BLINKAH DASHBOARD
Dashboard
Incident Heatmap
Meet The Team
Sign Out
Incidents
Safe Drivers
BLINKAH Units
License Plate Lookup:
SUBMIT


Dashboard
Incident Heatmap
Meet the Team
Sign Out
Cambridge
Brookline


BLINKAH DASHBOARD
BLINKAH Computational Frontend
Dashboard
Incident Heatmap
Meet The Team
Sign Out
Meet The Team
Parker Van Roy
Tech Lead & Client Liaison
Ritam Das
Creative Strategist & Visionary
Raghurama Bukkarayasamudram
Scrum Master
William Pine
Idea Guy
Ayush Upneja
Software Engineer