User Information Augmentation. Vision based Information Delivery System

Team: SDDec19-14

Advisor: Aleksandar Dogandzic

Client: Radek Kornicki of Danfoss

Problem Statement

- Operating heavy machinery can be dangerous
- Heavy machinery on highways can hinder traffic
- HUD to remedy this situation
- Determine viability on a larger scale
- Project serves as a proof of concept for the idea

Image source: https://github.com/FrenkT/LaneTracking

Related works

- University of Pennsylvania, gaze tracker robot
- Autonomous vehicles
- Google glass

Requirements

Functional:

- Identify important objects
- Track user's eyes
- Project a heads up display
- Verify user looked at important object
- User friendly calibration

Non-Functional

- Eye tracking and object detection in real time
- Detection done accurately
- HUD updated consistently
- System being portable

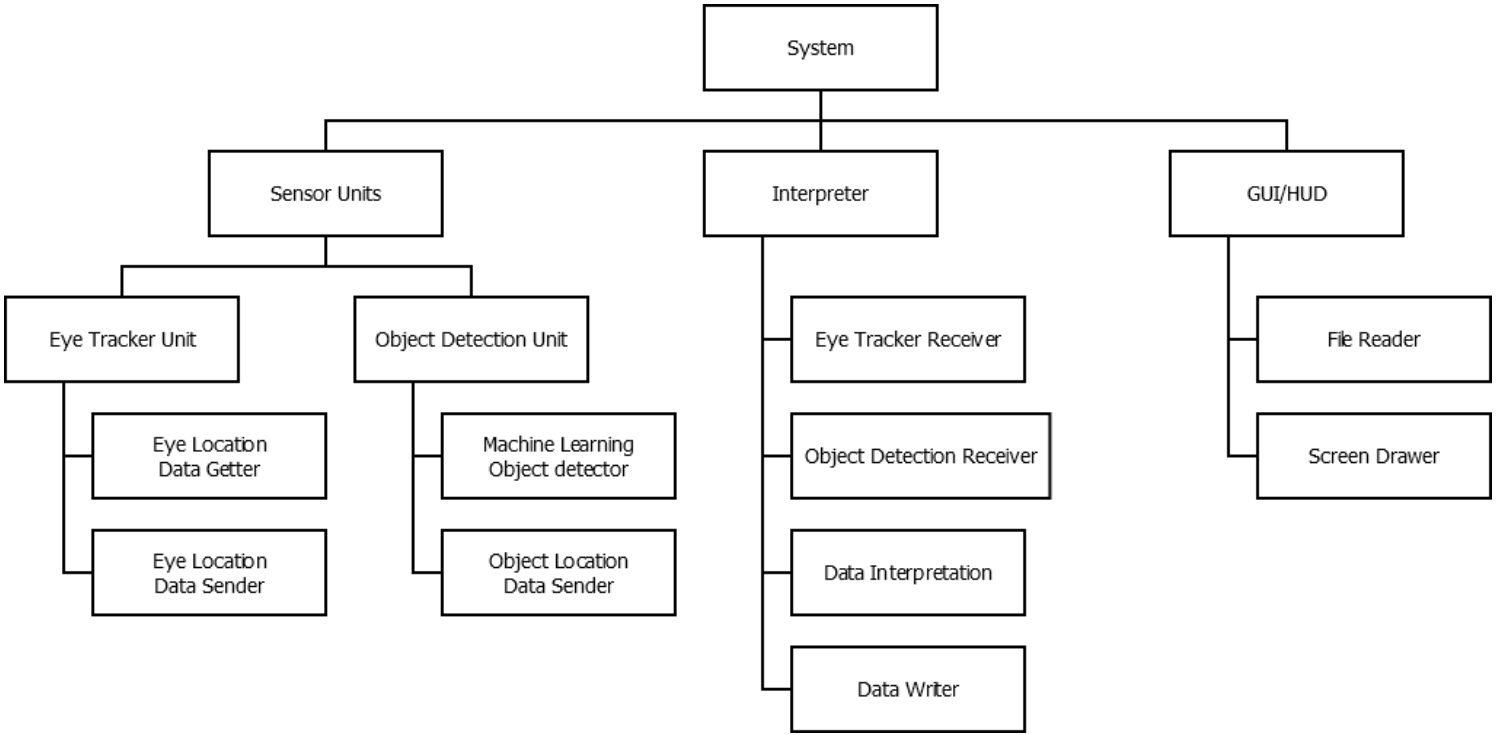
Other Constraints

- Economically Feasible
- +70% Transparent film
- Jetson TX2
- Tobii Eye Tracker 4C

Potential Risks and Mitigation

- Jetson’s speed
- Technical Experience
  - Tensorflow
  - Real-time systems
- Connectivity issues between hardware

Functional decomposition

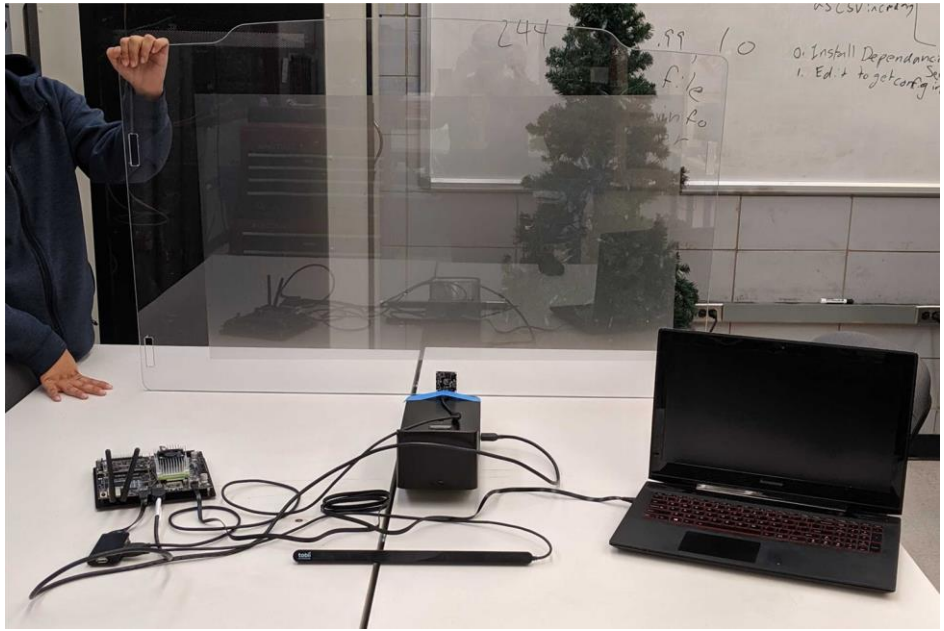


Demo video

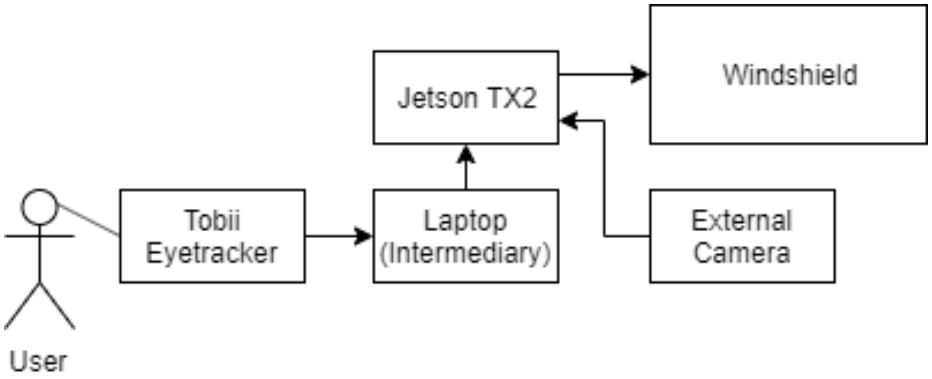
Demo video

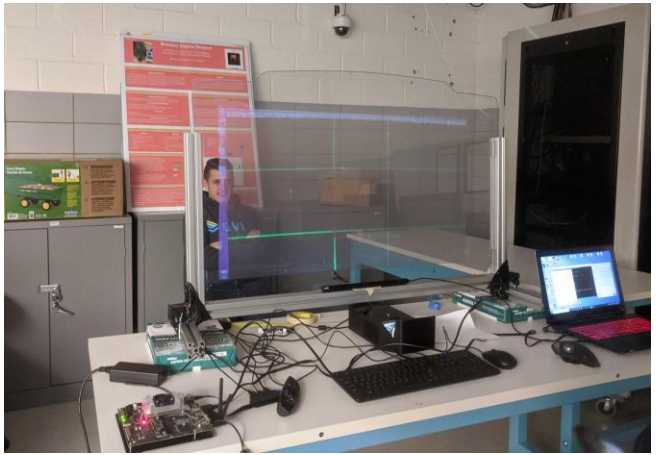
Devices Used

- Jetson TX2
- Tobii Eye Tracker 4C
- Projector
- Projection Film
- Windshield
- Camera
- Laptop

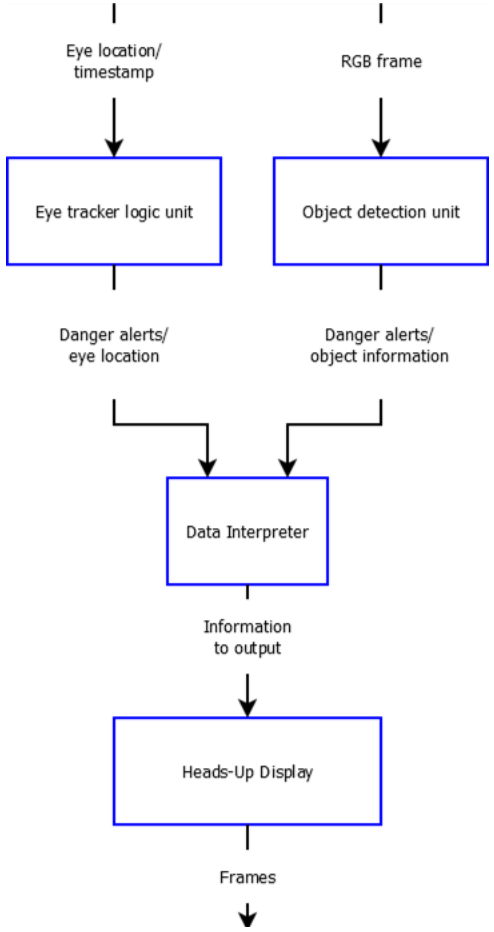


Hardware Layout



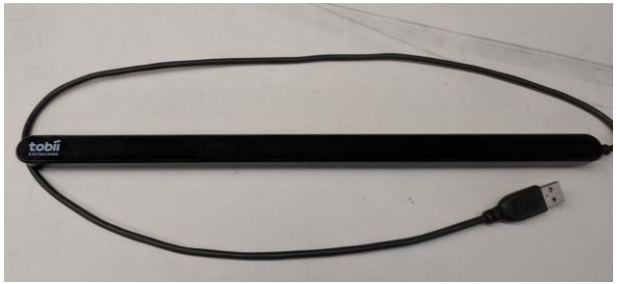


Software Layout



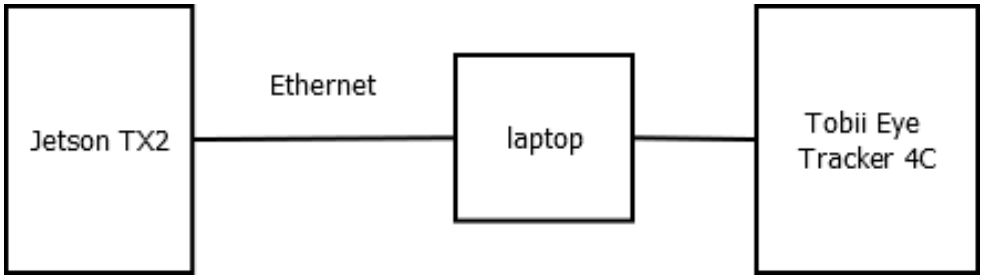
Eye Tracker Challenges

- Focused on Jetson TX2 and Tobii Eye Tracker 4C
- Drivers only for x86-64 system
  - WINE
  - QEMU
  - LattePanda
- Eye tracker calibration



Eye Tracker Design

- Intermediary device
  - Windows Laptop
  - Ethernet connection
  - UDP socket
- Constantly scans and sends timestamp, horizontal, and vertical

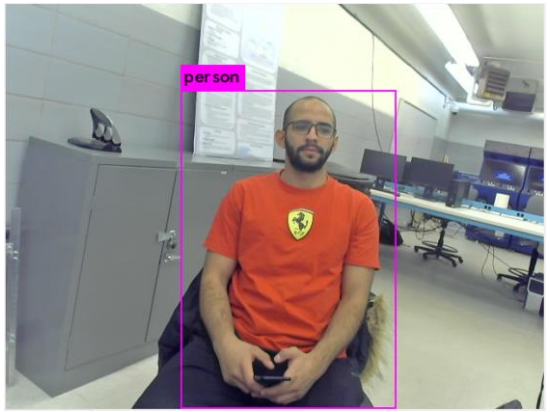


Object Detection Challenges

- Realizing objects at an acceptable speed
- Realizing all resources on the Jetson
- Realizing eye coordinates at real time

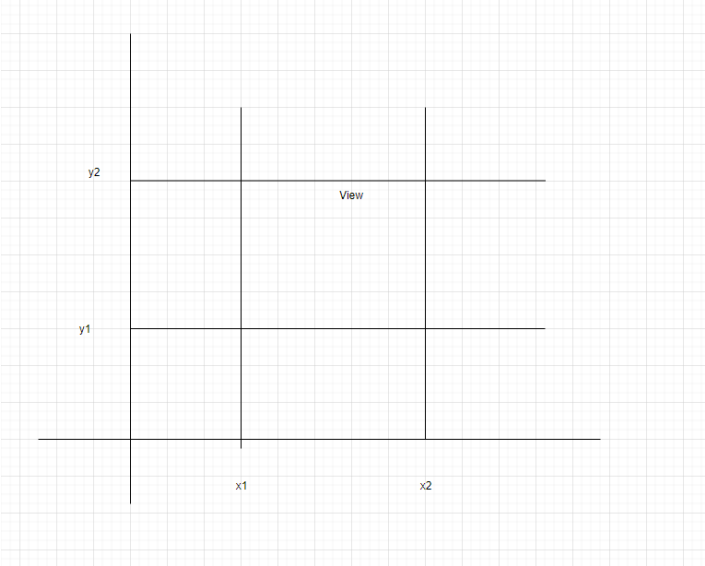
Object Detection Design

- Utilizes TinyYolo for object detection
- Grabs coordinates for each object
- Sends data to interpreter via UDP socket



Interpreter Design

- Two separate units for sensors
  - Connected via sockets
- Scales data
- Comparing eye location and object location
- Write to Json file list of objects



Heads-up Display Challenges

- Many alternatives that worked, but not well enough.
- Many ‘almost solutions’ found during research.
- Previous approaches: HTML w/ JS, Node.js, React

```
var canvas = document.getElementById('DHUD');
if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    while (1) {
        //Clear the screen to draw the objects
        ctx.clearRect(0, 0, canvas.width, canvas.height);

        await fetch(path, {
                mode: 'no-cors'
            })
            .then(response => response.text())
            .then(data => {
                console.log(data);
                objects = data.split('\n');
                objects.forEach(function(object) {
                    //Parse the object
                    var info = JSON.parse(object);
                    ctx.beginPath();
                    ctx.lineWidth = thickness;
                    ctx.strokeStyle = color;
                    ctx.rect(info.x, info.y, info.w, info.h);
                    ctx.stroke();
                });
            })
            .catch(error => console.error(error));
```

```
import React, { Component } from 'react';
import './App.css';
import { hot } from 'react-hot-loader/root'
import * as data from './testData.json';

class App extends Component {

  render() {
    return (
      <React.Fragment>
        {Object.values(data).map((object) => {
        console.log(object)
        return (
            <div style={{ position: "absolute", height: object.h,
        )
    })}
      </React.Fragment>
      );
  }
}

export default hot(App);
```

Heads-up Display Design

- Python to read JSON file
- Breaks data into objects
- Draws each object on the frame
- Easy to add new information

```
# Make a new frame
frame = np.zeros([1080, 1920, 3], np.uint8)

# Draw the detected objects on the frame
with open('testData.json', 'r') as datafile:
    data = datafile.read()


objs = json.loads(data)
for objNum in objs:
    x1 = objs[objNum]['x']
    y1 = objs[objNum]['y']
    x2 = x1 + objs[objNum]['w']
    y2 = y1 + objs[objNum]['h']
    frame = cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 10)

# Show the frame
cv2.imshow("projector", frame)
```

Other challenges

- Brightness of projector
- Projection film does not capture image well
- Jetson proved to be a bottleneck

Testing

- Unit testing for individual components
  - Each piece had to work on its own before being added to the larger system
- Manual QA once all pieces were together

Conclusion

- Client is happy with produced product
- Feasible to expand
- Setup not ideal

Future Work

- Expand what is shown to the user
- Include what the object is in interpretation
- Add lane detection
- Add calibration for each user as height changes perspective of HUD

Questions?