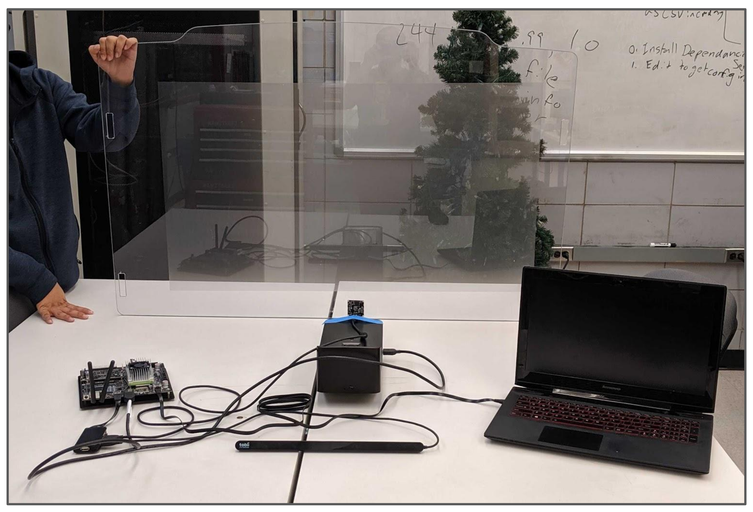
Senior Design December 19 - Team 14
User Information Augmentation.

Omar Abbas, Jonah Bartz, Aaron Michael, Dennis Xu
Client: Mr. Radek Kornicki of Danfoss
Faculty Advisor: Aleksandar Dogandzic

Introduction/Motivation

Operating heavy machinery has risks to both the operator and those around the machine. The User Information Information, Vision Based Delivery System is a proof of concept that aims to alleviate some of the risks that come along with the operation of heavy machinery. This system aims to ensure that drivers are alert and aware of their surroundings at all times through the use of a heads up display (HUD).



Design Requirements

Function Requirements:

- Able to identify important objects on the road(signs, people)
- Able to track user's eyes in the vehicle
- Project this in a heads up display style
- A system that will track eyesight over a large visual field

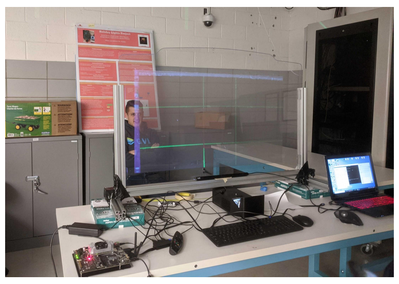
Non-Functional Requirements:

- Object detection and eye tracking to be done simultaneously at real time
- Object detection will to be done with an acceptable level of accuracy
- Fast, reliable and clear HUD capable of keeping up with user while they do the task at hand
- Reliable and secure module protected from malicious users and hackers
- A user-friendly calibration session for each user profile as pupils may differ due to different socioeconomic traits

Operating Environment:

As Danfoss deals with large machinery in a variety of different systems the operating environment can be assumed to be off highway environments in agricultural and construction vehicles. The system itself is intended to be mounted inside a vehicle with cameras mounted outside to collect data to display to the user.

Intended Users and Uses



Intended Users:
The intended users for our solution are primarily heavy machinery operators. Heavy machinery operators use vehicles such as combines harvesters, cranes, and bulldozers.

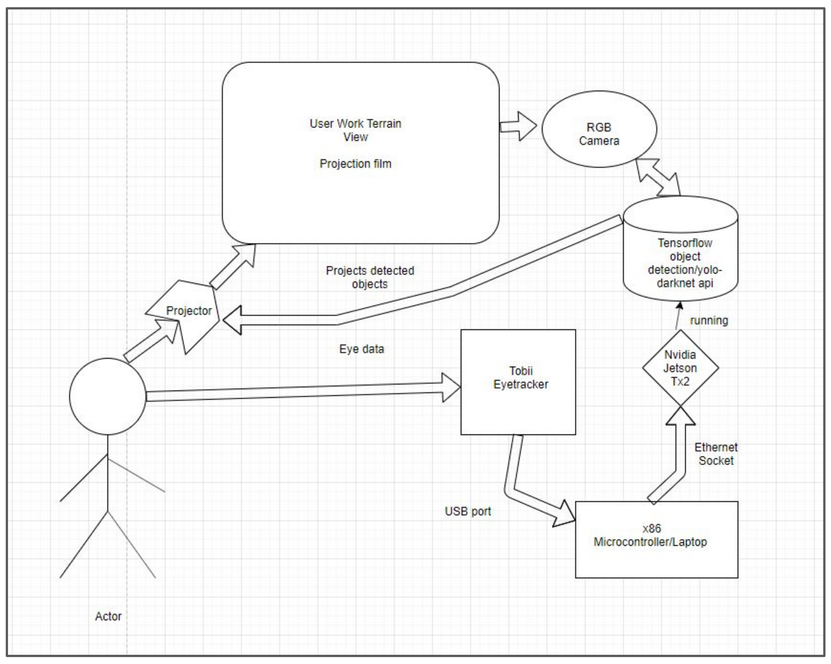
- Can't assume that the user has a lot of technical experience with troubleshooting software/hardware problems.
- User is busy controlling vehicle

Intended Use:
The intended use for our solution is improving safety when someone is operating heavy machinery.

- The solution needs to be fairly generic so it can be put into a lot of different machines without much modification.
- Easy to add functionality for different heavy machinery

Design Approach

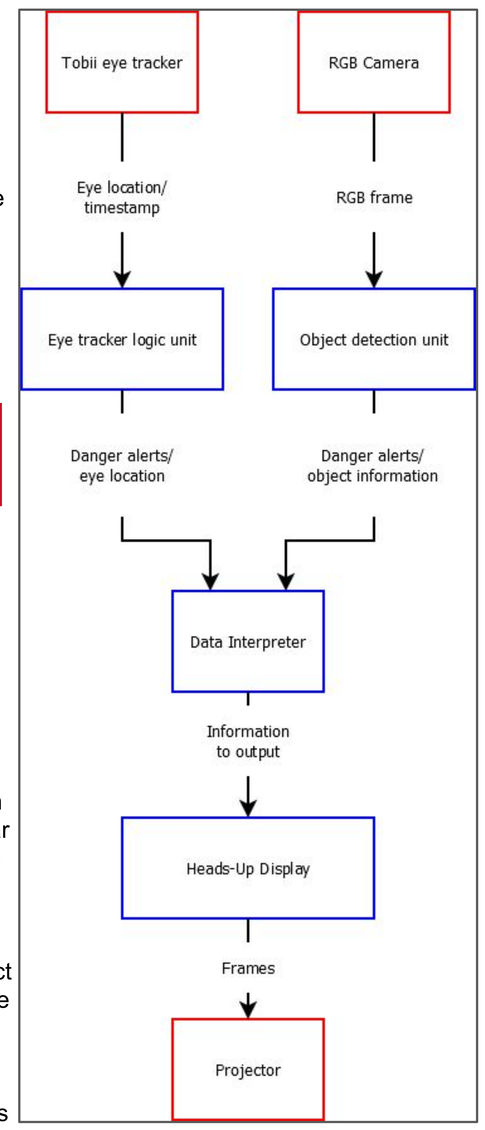
The block diagram to the below illustrates the topological view of the system.This is made up by:



- Tobii Eyetracker - determines where the user is looking
- RGB Camera - gets video from in front of the vehicle
- Projector - projects the HUD onto the windshield
- Microcontroller(intermediary device) - controls the eyetracker (Tobii drivers only for x86 systems)
- NVIDIA Jetson TX2 - run object detection, HUD, and interpreter code

The software block diagram can be seen to the right. The entire system is made up by:

- Eye tracker Logic unit - this retrieves the data from the eye tracker
- Object Detection unit - takes camera input and finds objects in a frame
- Data Interpreter - uses inputs from sensor logic units and decides if the user is paying attention to important objects
- HUD - takes info from interpreter and draws a screen



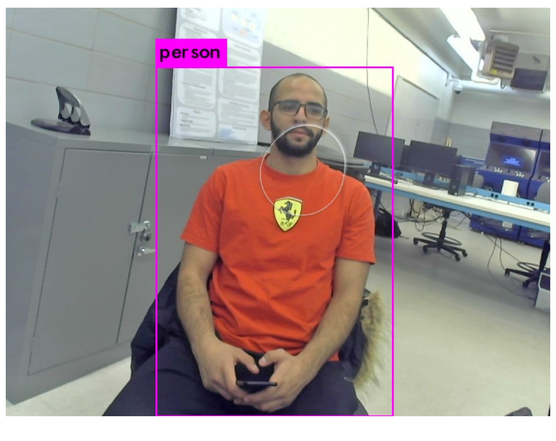
Technical Details

Object Detection: Run by *You only look once* API which utilizes tensorflow machine learning to detect objects. CUDA and openCV are utilized for real time webcam object detection, and a max of 22 fps is given. This is limited how much of its resources the Jetson can use and can definitely be improved with more low level Jetson configuration.

Tobii Eyetracker: This device utilizes a light source to illuminate the eyes giving highly visible reflections. Thus, coordinates of where we are looking at on a screen are captured and represented by a big bubble such as the one shown in the picture below. These coordinates are later compared with the box coordinates and if found that it is lined up, the box should disappear signifying that the user saw the box. An ethernet connection is also used to communicate between the eye tracker process and the interpreter process since they are on two separate machines.

Interpreter: This is written using base C++11 and takes input from the object detection and eye tracker to make decision on what should be drawn on the screen. It hosts a server sockets that it will reads sensor data from and communicates to the HUD using a pipe.

HUD: This piece of the project was created in JavaScript. This decision was made because it allowed for a lightweight solution and has the ability to tack on features easily due to its modualbility. The HUD was built in VS Code using React. The library React Hot Loader was used to allow for faster loading of data upon change.



Testing

We utilized different types of testing for our project. Each module of the system had unit testing applied to ensure that it worked with dummy data before being integrated into the overall system. Because we had many individual pieces of hardware we wanted to make sure each component worked individually before trying to connect them:

- Jetson object detection repeatedly tested for higher performance. Improved FPS drop when detecting 1 or more objects
- Tobii Eyetracker was calibrated for each of our members. It is not connected directly to the Jetson and so uses an intermediary device to monitor users eyesight on the user work terrain view. Multiple coordinates have been tested for this.
- Communication between modules of the system are always tested individually. Since we incorporate a great deal of sockets, it is vital that they are getting the information across to the right places. We mainly use UDP sockets since we are throwing lots of data at the modules at any given time.