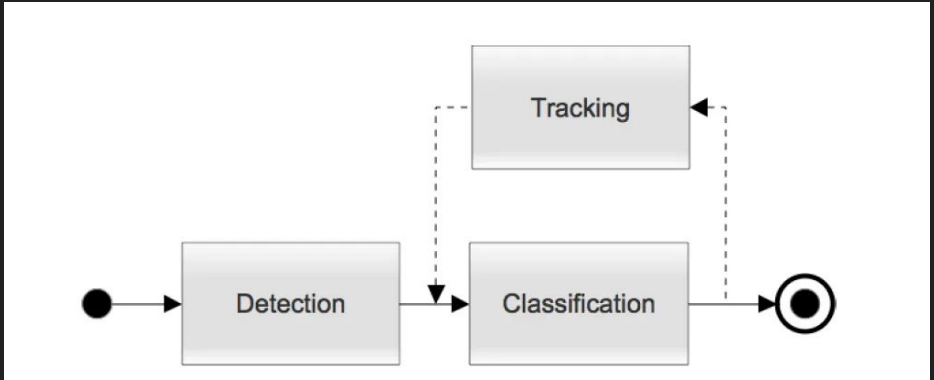
CDR – 2A1

Traffic Light Detection and Tracking

Morgan Roberts, Aaryan Shenoy, Clayton Gowan,
Xiaohu (Max) Huang, Robert Madriaga

Project Background

Background



The Problem

- Fast and reliable detection of traffic light, and light states
- Can be broken down into 3 parts
  - Detection
  - Classification
  - Tracking (loops back to classification)

Background

Traffic Light Detection Requirements

- Detection
  - Identify image
  - Deep Learning is a natural solution
    - Can be computationally expensive
- Classification
  - Lights must be classified by:
    - Type (arrow direction, solid, blinking, etc)
    - Color (red, yellow, green)
- Tracking
  - Faster than detection/scanning of the whole image
  - Conserve data between frames
    - Compensates for lack of data from detection

Needs Statement

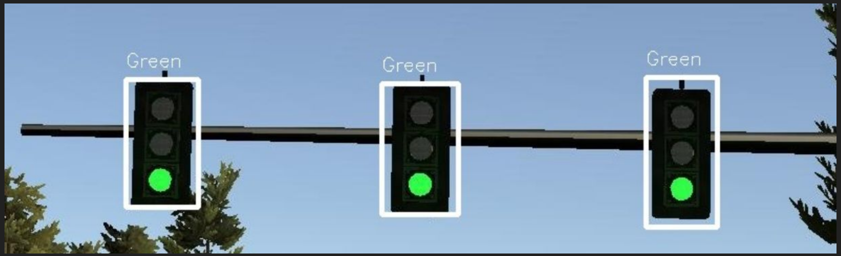
Needs Statement

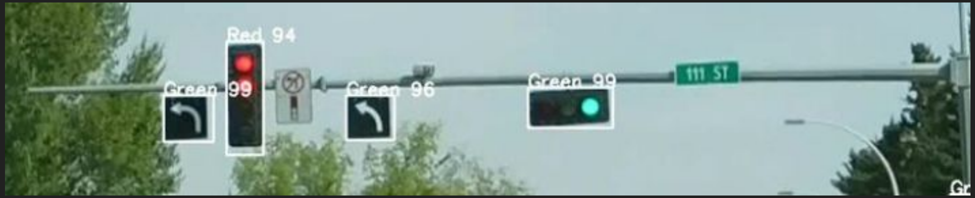
- Need – to detect, track and classify traffic lights
    - Prioritize speed and accuracy
    - Track lights across multiple frames
- Why – Self-driving vehicles must react to traffic lights.
    - Many systems use image data for this purpose.
    - These algorithms are usually to slow.
        - Reaction times are vital for safety

Goals and Objectives

Goals and Objectives

- Goal: Develop a speed-focused detection model for traffic light types and colors, analyze the states of the lights, track detection results across multiple frames, and integrate with ROS.
- Motivation:
  - Fast detection and tracking speeds are crucial in autonomous driving applications
  - Modularization of detection and tracking components

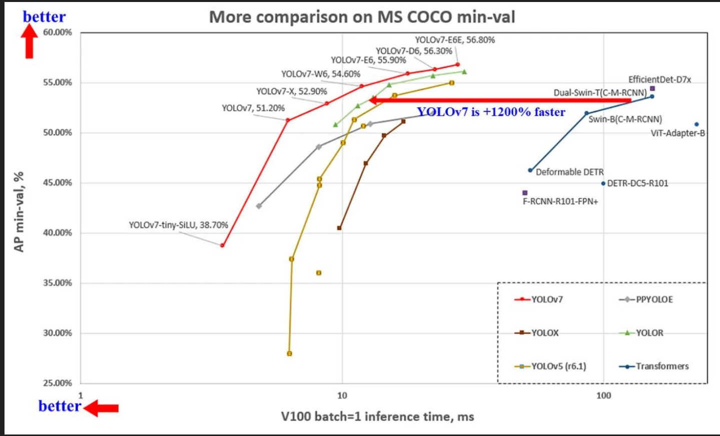


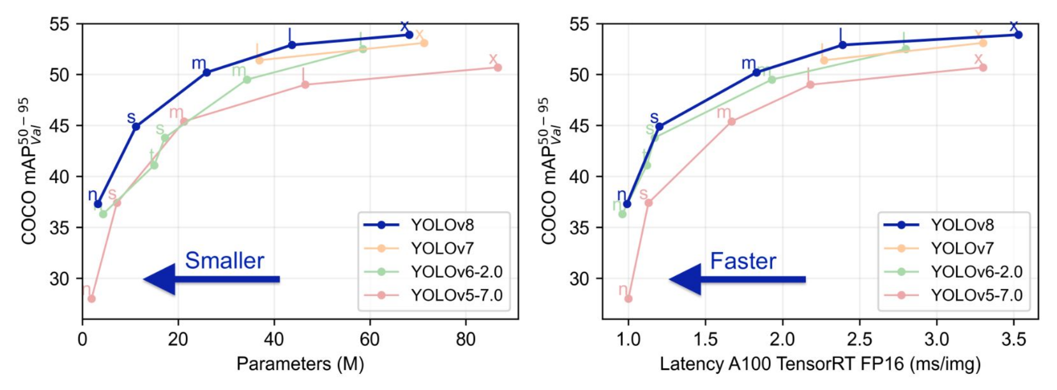


Design Alternatives

Detection Model

- We have chosen to continue using YOLOv8 due to findings indicating better detection performance and speed when compared to older YOLO versions and other alternatives.





Multi-Object Trackers

- DeepSORT
    - Popular tracking algorithm that utilizes frame trajectory predictions and deep learning
    - Uses YOLOv8 bounding boxes to predict location in successive frames
- Norfair
    - Open-source Python library that supports object tracking in moving cameras
    - Compatible with YOLOv8 model through use of its bounding boxes
    - Use of Norfair is promising compared to DeepSORT
        - Good Performance
        - Ease of implementation
        - DeepSORT issues

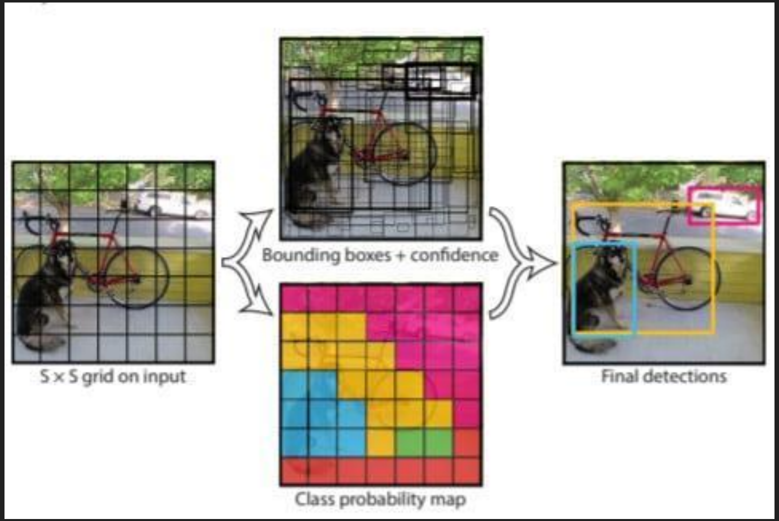
Proposed Design

Proposed Design

- Input: ROS Bag Datasets
- Output: Camera frames with detected traffic lights labeled
- Tools to Use:
    - Machine learning framework
        - YOLOv8 detection model
        - Norfair library
    - ROS Bag Screenshot Dataset
    - ROS node system

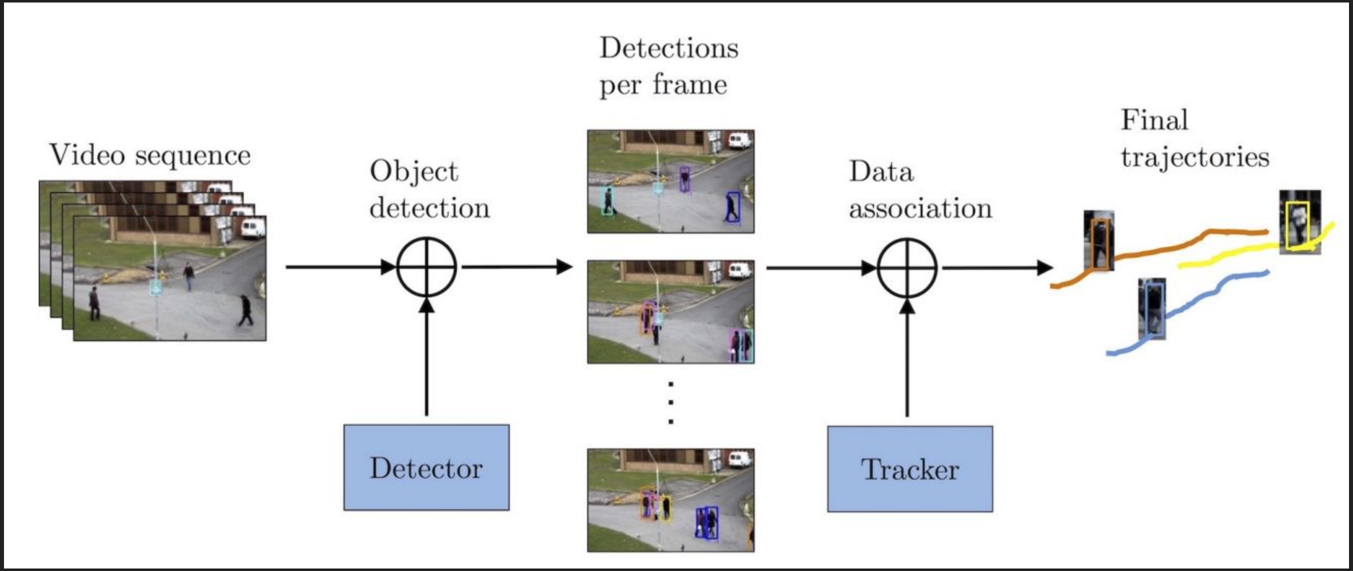
Proposed Design

- YOLOv8 detection model
  - Runs on each camera frame to produce detections



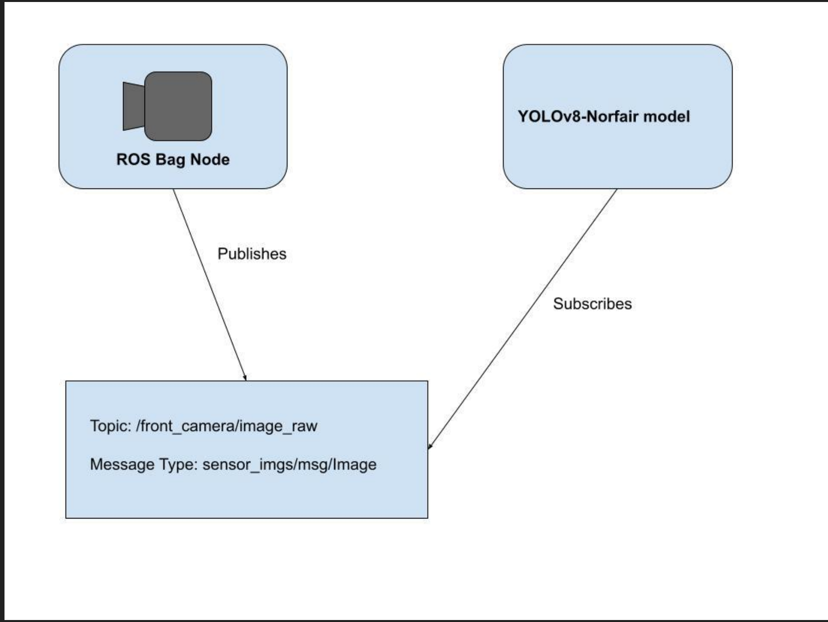
Proposed Design

- Norfair Python library
  - Library functions utilize YOLOv8 bounding boxes to track traffic lights across frames



Proposed Design

- ROS and machine learning integration via publisher-subscriber model



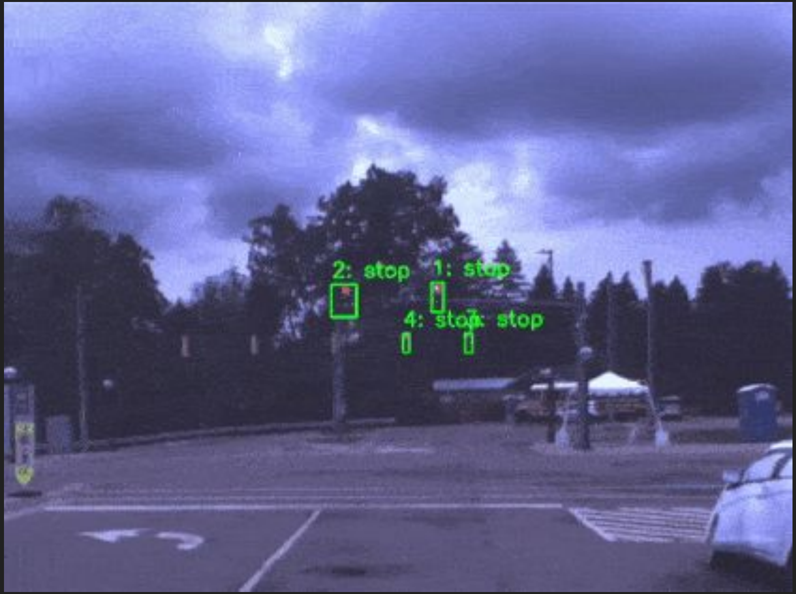
Vision for Resulting Product

Current Progress

- ROS node network is set up
  - Contains processes that play camera footage and run YOLO and object tracking models
- Python program created to process each input frame through YOLO and object trackers
- YOLO/Norfair model bounding boxes detects lights better than YOLO/DeepSORT due to default DeepSORT feature extractor trained on humans

Current Progress, Cont.

Detection



Tracking

Note: Video is played at 3x actual speed

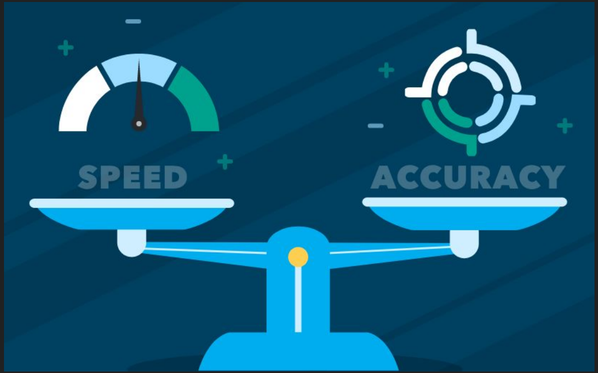
End Result Vision

- Take camera feed as input
- Have the YOLO model detect and classify traffic lights accurately and quickly
- Use detections to continually update the tracker, which will assign IDs
- Filter the tracked objects to only include the relevant traffic lights
- Determine flashing light states
- Output the currently tracked traffic lights (amount, type, etc.) in the desired output

Approach for Design Validation

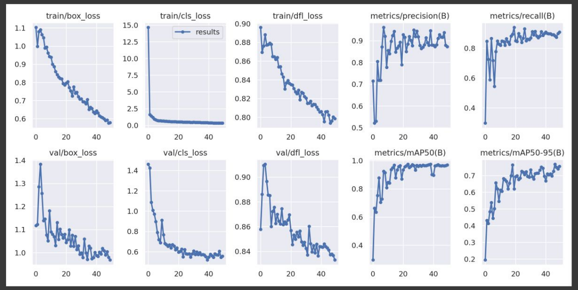
Approach for Design Validation

- Ensure we are running at an adequate framerate with good detection speed while also maintaining a high level of confidence and accuracy in detection
    - Measurements: FPS, detection model confidence levels
- Find an optimal balance between model size, speed, and accuracy
    - Compare pre-trained models/different datasets with custom models
- Verify that multiple objects can be tracked accurately across frames
- Make sure pipeline functions correctly with desired formats/outputs



Approach for Design Validation, Cont.

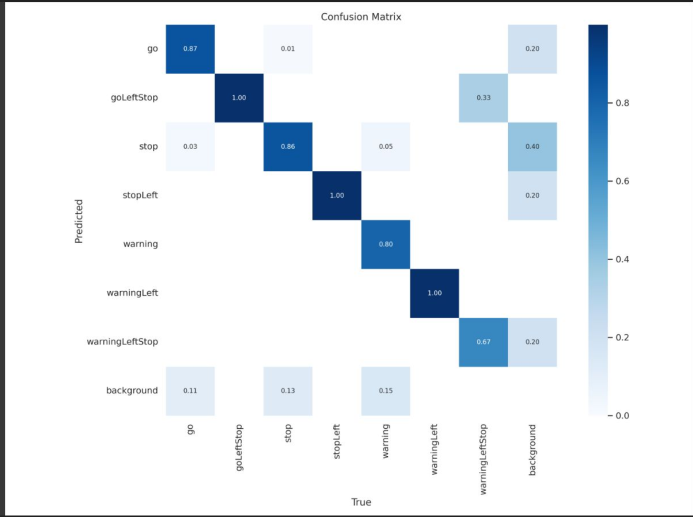
- Utilize YOLO training output to compare model performance on testing and validation data, observe inference speed and performance on rosbag files
- mAP, Recall, etc.




```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      13.8G     0.5782     0.3309     0.7987         46       1280: 100% 46/46 [01:10<00:00,  1.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 3/3 [00:09<00:00,  3.21s/it]
                   all         69        176      0.873      0.908      0.969      0.756

50 epochs completed in 2.149 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 22.7MB
Optimizer stripped from runs/detect/train/weights/best.pt, 22.7MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.53 🚀 Python-3.9.16 torch-1.13.1+cu116 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11128293 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 3/3 [00:08<00:00,  2.91s/it]
                   all         69        176      0.915      0.884      0.965      0.755
                    go         69         75          1      0.775      0.958      0.661
            goLeftStop         69          4      0.914          1      0.995      0.895
                  stop         69         71      0.944      0.718       0.93      0.593
              stopLeft         69          1      0.619          1      0.995      0.895
               warning         69         20      0.996        0.8      0.885       0.73
           warningLeft         69          2      0.933          1      0.995      0.798
       warningLeftStop         69          3          1      0.895      0.995      0.708
Speed: 6.2ms preprocess, 28.2ms inference, 0.0ms loss, 2.0ms postprocess per image
Results saved to runs/detect/train
```



Economic Analysis/Budget

Economic Analysis/Budget Update

Each team gets a $500 Budget, and out of our budget we spend:

Human Labor - $0

Software - Open Source - $0

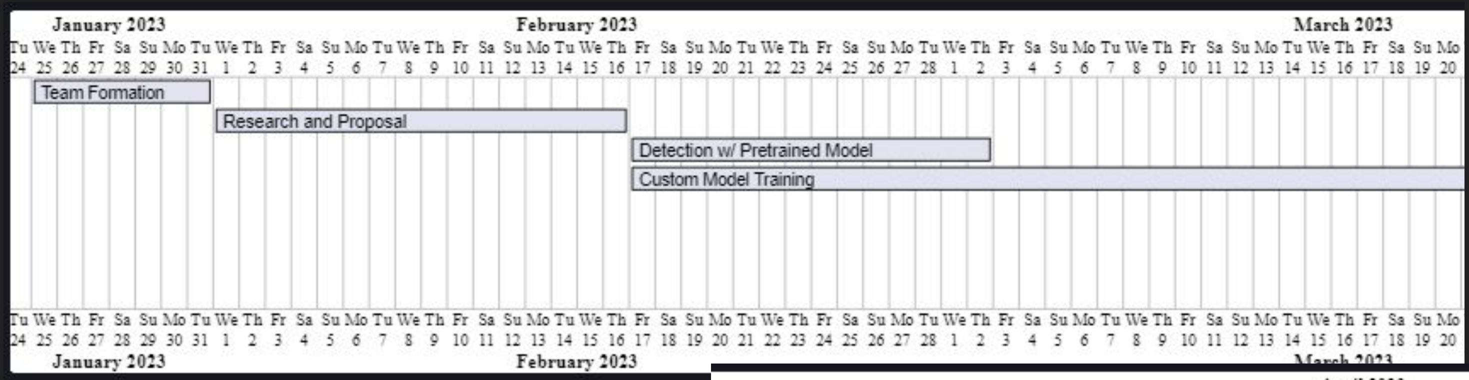
Software - Closed Source / Cloud Services - $variable

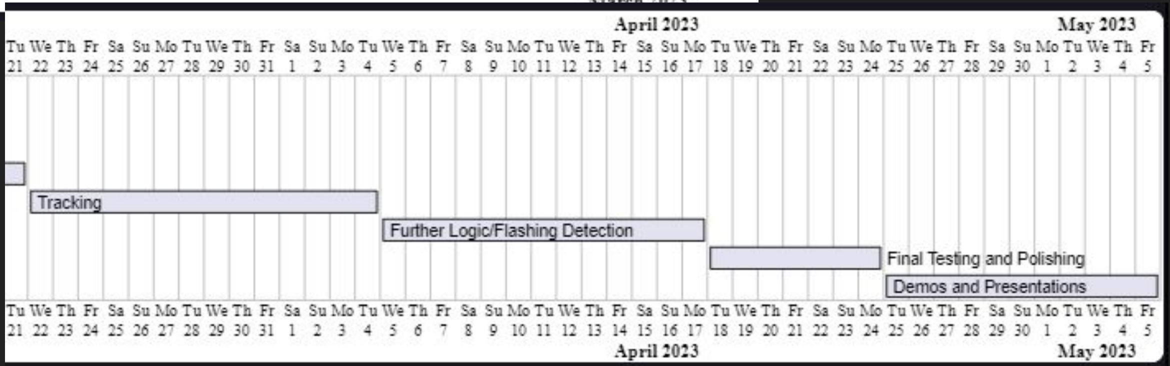
- Money could be spent on an upgraded Colab subscription
- This has not been necessary so far

Schedule of Tasks

Gantt Chart

- Estimation of development schedule (subject to change)
- Model Training took longer than expected





Project Management and Teamwork

Teamwork and Roles

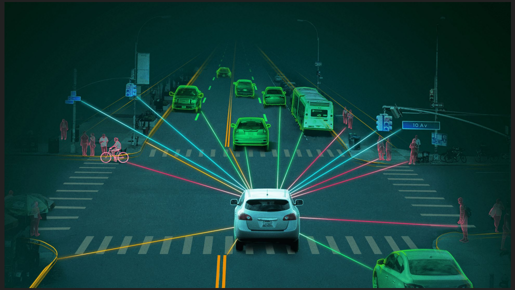
- Morgan Roberts: Team Leader
- Aaryan Shenoy: Systems Design
- Clayton Gowan: Software Design and Model Training
- Xiaohu (Max) Huang: Software Design and Testing
- Robert Madriaga: Technical Writing and Dataset Management

- There have been weekly in-person team meetings on Tuesdays/Thursdays and Discord meetings as needed
- Members may assist in other roles/responsibilities as needed

Societal, Safety, and Environmental Analysis

Societal, Safety, and Environmental Analysis

- Autonomous vehicles have the potential to improve convenience and quality of life.
  - Optimal travel times
  - Increased accessibility to transportation
- Privacy Concerns
  - Requires monitoring equipment
- Safety Concerns
  - Even minute errors in software or hardware can be catastrophic
  - Lead to loss of life and property damage
- Safety Benefits
  - With ideal monitoring, computation, and decision making, a self driving car can react to and avoid dangerous situations faster than humans and with more consistency
- Environmental Benefits
  - Effective navigation leads to a reduction in emissions



Manufacturability

- **Software Compatibility**
  - **Software is compatible with Linux operating systems that is able to run ROS2 Foxy and Python 3, or though a VM box or docker environment**
  - **Lack of support or updates to any piece of software used may cause errors as features may be deprecated or incompatibility with newer versions**

- **Hardware Compatibility**
  - **Designed to work with high resolution camera at 60 fps**
  - **Requires moderately powerful modern hardware to run on**
  - **Higher frame rate cameras may cause issues with blinking light detection code**

Questions