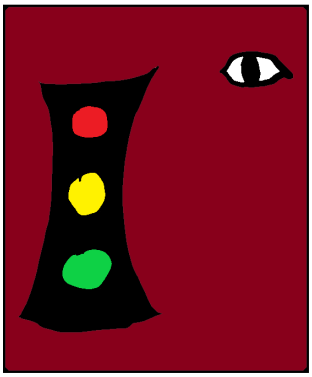
2A1: Traffic Light Detection and Tracking



Project Proposal

2A1

Aaryan Shenoy
Clayton Gowan
Morgan Roberts
Xiaohu Huang
Robert Madriaga

Department of Computer Science
Texas A&M University

02/14/2023

Table of Contents

1 Executive summary (1 page; 5 points)

Traffic lights are a ubiquitous tool for the drivers of vehicles all over the world. They offer a method by which to regulate and sustain the flow of traffic, so their importance cannot be overstated. As we move into the age of automation, it has become clear that traffic light detection will be a vital ingredient for the success of autonomous vehicles. After all, in order for self-driving cars to function properly, they need to be able to detect, react, and behave as a rational driver would in all situations, and this conduct is not possible without auxiliary knowledge of the state of traffic lights. Indeed, as it currently stands, there already exist many implementations of traffic light detection. Most if not all of these require camera data, and some supplement with data from other sources such as lidar and local maps. Unfortunately, lidar is only effective at close range, so its use with traffic light detection is naturally limited. The primary solution is to use machine learning algorithms on the camera data to detect and classify traffic lights. However, driving is an environment in which reaction time and accuracy are second to none in terms of importance, and a neural network needs to be as responsive and correct about its predictions as possible. Current implementations are often either too slow or inaccurate to be considered a viable solution. Therefore, we seek to develop a method of traffic light detection, tracking, and classification that prioritizes confidence in recognition results and is quick and responsive.

We will simulate this task of traffic recognition in an ROS node system. Our inputs are ROS Bag file datasets, consisting of files that represent camera feeds that depict scenes of driving in urban environments. Our outputs will be in ROS Bag file format and will yield camera frames with all displayed traffic lights automatically labeled with relevant information such as signal color. We have decided that the YOLOv8 detection model and the DeepSORT algorithm will consist of our machine learning framework. The framework will be trained on the provided LISA Traffic Light Dataset and the framework will be incorporated into an ROS node system. Our machine learning framework will assign the job of traffic light detection and tracking to the YOLOv8 model and DeepSORT algorithm, respectively. The reason why we want to use the YOLOv8 detection model is because it is the fastest model for object detection in individual camera frames. Additionally, we choose to utilize the DeepSORT algorithm because it is regarded as a highly effective multi-object tracking algorithm that is compatible with YOLOv8. DeepSORT will aid YOLOv8 in traffic light recognition because object tracking will help us avoid cases where the sole use of YOLO would stop detection of traffic lights hidden by surrounding elements or traffic lights that are blinking. YOLOv8 will output bounding boxes locations that DeepSORT will use to keep track of detected traffic lights in all frames until the object moves out of frame. We will encompass our machine learning model and our ROS bag files within two separate ROS nodes. The ROS nodes will communicate with each other in a publisher-subscriber ROS model. The publisher node will be the ROS Bag file node and it will send its camera footage to the machine learning node, which is a subscriber node. The machine learning node will also be a publisher node that sends ROS Bag file results containing frames with bounding box detections to other subscriber nodes.

Overall, our goal is to develop a model that can detect, track, and analyze the state of traffic lights. The output should be produced in the correct format and responsive enough to give the other components of the autonomous driving system time to react to stimuli. Traffic light detection is an essential component of the autonomous vehicle decision making pipeline, and can be considered the entry point. If this implementation functions correctly, then the result is a safer and better-performing self driving vehicle. Another objective for our team is to become better acquainted with machine learning and practical software such as ROS. Also, we hope to become better team members and develop enhanced collaborative skills.

2 Introduction (1 page; 20 points)

2.1 Needs statement (5 points)

Self-driving vehicles require a means by which to detect traffic lights. Many detection systems use cameras and neural networks to detect and recognize traffic lights. However, in an environment where reaction times are integral to the safety of the passengers, these algorithms are far too slow to be of practical use. We need to develop a means to detect traffic light colors and types and track these detections which prioritizes speed while still maintaining a high level of accuracy.

2.2 Goal and objectives (10 points)

The goal of this project is to design a neural network which is able to take an image and accurately detect any traffic lights along with their color, type, and location as fast as possible. The data will be output as an array of these values. Ideally, this process should be extremely fast and responsive. Our first objective is to get familiar with the tools and libraries, such as ROS, YOLOv8, and DeepSort, as we plan to use these extensively. Next, we aim to test detection of traffic lights using YOLOv8 and downloading a pre-trained model to test the functionality. We will then use ROS and YOLOv8 to train our own model and derive weights for it. Next, we will configure DeepSort to track lights across frames to reduce workload and frequency of the model. It is important that we are able to detect special cases such as flashing lights, as the state evolves over time, and that will be our next objective. Our final objective is to combine our previous objectives together into a pipeline using ROS and to polish our design to increase efficiency and accuracy. It should be noted that we have no plans to make purchases for this project, as we are designing only software and will use our own hardware for testing and development.

2.3 Design constraints and feasibility (5 points)

In regards to constraints that affect the practicality of our project, there are several that must be addressed. There exists an inherent time constraint, as we have set due dates. We also must spend time training the model, and this limits the amount of time that we are able to spend perfecting the project. It should be noted that not every member of the team is familiar with machine learning, which plays an important role in this project. What's more, the entire team has little familiarity with the active software such as ROS, DeepSort, and YOLOv8, and this imposes a technical constraint on the project. Our team is working on laptops with very apparent hardware constraints. Factors such as the GPU and CPU can directly affect the speed of the model, and this should be taken into account. The model must also meet certain requirements, such as a desired level of confidence. Our implementation must also output to ONNX format as an array of traffic light data.

3 Literature and technical survey (1-2 pages; 10 points)

Project 1: Traffic Lights Detection and Recognition Method Based on the Improved YOLOv4 Algorithm [1]

The aim of this project was to optimize the YOLOv4 detection model in order to improve its ability to identify small objects and to refine its tracking precision. Modifications of feature extraction network layers and calculations of uncertainty of bounding box coordinate measurements were utilized to enhance the model's general accuracy and detection of small objects. Experiments were performed on LISA and LaRa traffic light datasets to analyze the performance of the study's algorithm along with 5 other neural networks. The study's YOLOv4 algorithm resulted in the highest performance based on AUC and mAP metrics. The authors concluded that both its network modifications and uncertainty calculations improved detection accuracy; however, the network modifications increased computations in their algorithm that resulted in negligible increase in runtime. Authors proposed that research into trajectory predictions of detected traffic lights be conducted as an extension to their study. This study is relevant since it overcomes a weakness of the YOLO model and highlights in its extension proposal that object tracking

should be incorporated into YOLO detection. The proposed design will attempt to exploit the potential of object tracking in our machine learning model.

Project 2: A YOLO Based Approach for Traffic Light Recognition for ADAS Systems [2]

This project aimed to develop a means to detect and classify traffic lights in advanced driver assistance systems in order to alert drivers when they may cross a red light thus helping to avoid potentially deadly situations. This was accomplished by training a model using the YOLOv4 model on the LISA dataset which includes images of traffic lights from the streets of California. First, the DNN model was trained to identify traffic lights given a whole frame of input. The output here would then be sent to another module to estimate distance and decide whether or not to notify the driver. In addition, there is another step which filters out traffic lights that are not in the driver's path so the system will not alert the driver to irrelevant traffic lights. Three classes, green, yellow, and red were trained in the model and YOLO was found to be the most reliable algorithm. This is partly due to its ability to detect without any backpropagation. The alert system was done by estimating the rate of change of distance to the light and combining that information with data on whether the driver was slowing down or not and the actual distance to the light. Every other frame was processed instead of every frame to improve performance. The system had an average framerate of 13.4 with relatively high average precision results in the 90 percentage range, except for yellow in the 80s due to low training data. They stress that the accuracy of their model can be improved by including more images of yellow and horizontal lights which they were lacking. The researchers also stressed the importance of frame processing before detection in finding an optimal resolution. They settled on 608x608 since darknet requires a multiple of 4 and it struck a balance between speed and accuracy. It is important to choose a resolution that is not too high or too low.

Project/Research 3: Traffic Light Recognition — A Visual Guide [3]

This project describes the traffic light detection problem and shows techniques that can be used for efficient real time solutions.

The traffic light detection problem can be broken down into 3 parts:

- Identifying Regions of Interest
- Training a Classifier
- Tracking and Optimization

And there are several techniques that can be used for an efficient solution:

- Sliding Windows - slow brute force method that isn't particularly useful
- Cropped Sliding Windows - same as above but with regions of interest cropped out, faster
- Color Thresholding - fastest method, process image to show only red green and yellow
- BLOB analysis - pixel clusters obtained after color thresholding can be analyzed further by their shapes and sizes

Project 4: HDTLR: A CNN based Hierarchical Detector for Traffic Lights [4]

This project introduces HDTLR, a convolutional neural network based on DeepTLR which uses a hierarchy of object classes to classify an object. HDTLR is compatible with many different common feature extraction networks such as AlexNET, and is capable of handling different input sizes after training. The last layers are a multitask network split between region classification to create a probability map for each class and the second task being bounding box regression for each region similar to DeepTLR. The Softmax layer from DeepTLR is modified to use a tree that defines the hierarchy of objects, which determines the probability of each object's classification. The final decision is made from a combination of hierarchy, DeepTLR clustering.

Project 5: A deep learning approach to traffic lights: Detection, tracking, and classification [5]

This paper emphasizes the need for a system which can detect traffic light data without the need for map-based information and lidar data. The paper details an approach using deep learning, stereo vision, and vehicle odometry to detect, track, and classify traffic lights. Since the authors imposed limitations on their solution that coincide with ours, their proposed pipeline is something that we can use to similar effect. This paper helped to inform our decision to use YOLO for detection and classification, but to find another tool for tracking. Our proposed solution/design is very similar to some of these projects discussed above particularly in the use of YOLO for detection. This is a very popular algorithm for these kinds of projects and has been shown to be very fast. Our implementation is seeking to use a newer, possibly faster, version of YOLO than those used in some of the projects discussed above. Additionally, the use of a tracker in project 3 is also similar to our proposed design and need. We are seeking to use a multi-object tracking that not only uses Kalman filters and other motion-based algorithms but also works with deep learning to track objects.

4 Proposed work (35 pts)

4.1 Evaluation of alternative solutions (1 page, 10 points)

This is a critical aspect of your proposal. For any goal there are likely many alternative solutions. In most cases, the alternative solutions will emerge from your literature and technical survey. What you have to do here is analyze the pros and cons of each of these solutions (and hopefully additional solutions you come up with), and justify your decision to opt for a particular solution. As a guideline, this section should include *not less than five alternative solutions*:

- Alternative solution 1: With less frequency, apply filters to the image to get a grayscale image. Use the large YOLOv8 model on the entire grayscale to detect hitboxes. Feed hitboxes into deepsort tracker to track hitboxes across frames and detect state data such as blinking. Feed hitboxes into the small YOLOv8 tracker to get the color of light and the type data.
- Alternative solution 2: Tracking Techniques: One of the project requirements is to not only detect the traffic lights, but track them across frames as well. Tracking is different from detection in that instead of simply identifying an object on a single frame, tracking algorithms seek to predict the position/trajectory of an object across consecutive frames and may help account for dropped detection frames, occlusion, etc. There are different routes one can take in order to track objects. One is to use single-object tracking such as those included in the cv2 library including MOSSE, KCF, and more. These algorithms are simple to implement and can be very fast and effective, but are not as useful when multiple objects or traffic lights need to be tracked at once. This brings us to multi-object tracking algorithms such as SORT and DeepSORT. These algorithms are similar, but DeepSORT offers some key improvements over SORT. SORT does an acceptable job, but suffers from frequent id switches and cannot handle occlusion very well. DeepSORT seeks to improve on this by not only taking into account object motion but appearance as well. For these reasons, we will aim to implement DeepSORT. [6]
- Alternative solution 3: Optimizing YOLO models: The YOLOv4 detection model has been optimized in a study to improve the model's to detect smaller objects [1]. Two strategies of tweaking YOLO's convolutional neural network and applying mathematical models to calculate the uncertainty of bounding box location calculations were applied. The first strategy involved technical skill of modifying network layers through splicing and upsampling, while the second strategy implemented complex statistical knowledge of Gaussian distributions. These strategies were proven to enhance the detection accuracy of small and general targets based on concrete metrics like mean average precision (mAP) and area under Precision-Recall curve (AUC). However, implementing this solution requires expertise in machine learning and statistics that our

team does not have. Additionally, it is not known whether this strategy could be applicable to a newer model version like YOLOv8.

- Alternative solution 4: Mini-YOLOv3: Mini-YOLOv3 is an object detector that is optimized for embedded applications, it has slightly reduced detection accuracy but considerably less demanding on the hardware platform [7]. The parameter size of Mini-YOLOv3 is only 23% of YOLOv3 and achieves comparable detection accuracy as YOLOv3 but only requires 1/2 detect time. This may allow for better performance on the non specialized hardware we have access to (laptops), but the reduced detection accuracy may lead to lower detection reliability of the system.
- Alternative solution 5 HDTLR: Hierarchical DeepTLR model: In the paper by M.Bach et al., they describe how their model uses an alternative CNN approach to that of YOLO based on DeepTLR, and its capabilities to quickly identify traffic lights and their associated lanes [8]. Using their described approach we would need to choose a separate classification network such as AlexNet, and modify the DeepTLR Softmax function to use a hierarchy tree. Some of the advantages of this approach would be that HDTLR would be able to work with different input sizes from what it was trained on. Giving more flexibility but being more complex than the more commonly used YOLOv8.

4.2 Design specifications (3-4 pages, 20 points)

The project will detect traffic light boxes and track their light status through the YOLO detection model and the DeepSORT algorithm. Additionally, we will integrate our novel machine learning model into the Robotic Operating System (ROS) framework in order to utilize image processing services and contribute our algorithm to the framework. Model training and experiments will be run on camera datasets provided by the professor.

The ROS architecture will encompass our project's design as multiple agents like camera data streams and traffic light detectors are represented as ROS nodes communicating with each other [9]. In ROS, each node is a process that is able to communicate with other nodes through interprocess communication and contain computational logic for aspects like traffic light detection. The project's nodes will relay results of their computations to each other by acting as "publishers" that send messages of their respective result type to "subscriber" nodes. We will employ this form of ROS interprocess communication, known as topics, because we want to have input and output streams between our system nodes for continuous data processing.

Our project's model training input will be from the LISA Traffic Light Dataset. The LISA dataset consists of hundreds of thousands of JPEG camera frames displaying scenes from driving [10]. The camera data is sourced from a variety of urban settings differing in their light and weather conditions. We will train our detection model using parts of this database. Traffic lights will be labeled in the dataset's frames using the Roboflow tool. Our YOLO model will be configured to train with three primary classes in mind: green, yellow, and red traffic lights. This can also be expanded to specify arrow lights and other types of lights.

The system will run on a ROS bag file as input for traffic light detection and tracking. ROS bag files will be housed within an ROS node and its contents simulate a camera feed that depicts driving. The camera input node will serve as a publisher of the camera/video image data topic, which our machine learning model node will subscribe to. This file format is our model's input because it stores ROS message data that publisher and subscriber nodes can easily communicate with.

We aim to create a machine learning model consisting of the YOLOv8 detection model and DeepSort object tracking algorithm. This model will execute within a separate ROS node that will be linked to the ROS bag file input node through topic subscription. Machine learning in our model will assign our two main tasks of traffic light detection and tracking to YOLOv8 and DeepSort, respectively. It will output the same frames which are now marked with boxes that detect traffic lights and label their signals such as green, red, and yellow. The model's ROS node will be a publisher for these annotated frames that other

ROS nodes can subscribe to as shown in Figure 1. We believe the utilization of an object tracking algorithm like DeepSort will be efficient towards accomplishing our goal since detection of different light types does not have to occur on every frame.

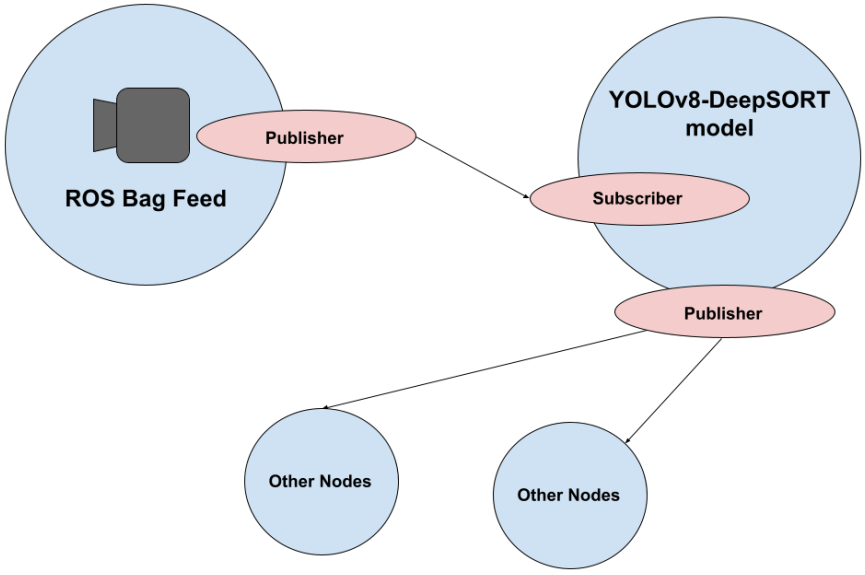


Figure 1: High-level project architecture

The YOLOv8 model is the latest and fastest version of the YOLO algorithm. We chose YOLO for object detection because it is regarded as the best model for this task. The model will take an input frame and divide it up into $S \times S$ grid cells [11]. Each of these cells has the ability to mark the image with B bounding boxes. S and B are pre-set numbers for the model's configuration, while a bounding box is a rectangle that attempts to fully encircle a detected object in the image. YOLO requires a grid cell to detect an object if the object's center is within the cell. For a bounding box, the probability that the box contains a relevant object, location coordinates of the box's center, dimensions of the bounding box, and the probabilities that the associated object belongs to each respective class that we are trying to detect. In order to get rid of redundant bounding boxes out of the initial B boxes, ratios of the intersection area of two bounding boxes over their combined area are calculated and boxes with high resulting values are kept. Finally, a step called non-max suppression selects boxes with the highest probabilities that they are an object out of the remaining boxes. The algorithm is visualized in Figure 2 [11].

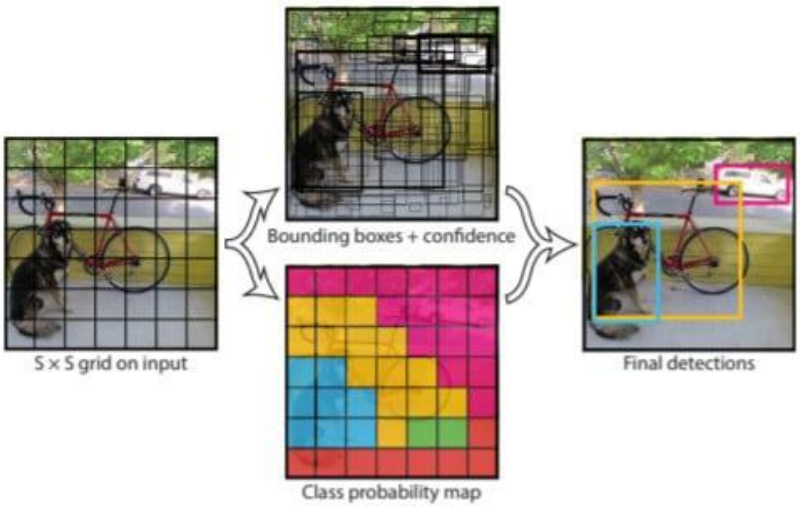


Figure 2: YOLO grid partition and bounding box prediction

The next component of the machine learning architecture following detection will be a multi-object tracker. It is critical for object tracking to be developed in our model because real-world cases involving the detected object being hidden by other surrounding elements would hinder the effectiveness and efficiency of a model that only relies on detection. We seek to integrate DeepSORT object tracker because it is a highly effective tool that has compatibility with the YOLO model. DeepSORT builds on an algorithm called Simple Online Realtime Tracking (SORT), which has four stages [12]. It first detects an object using YOLO, next it computes the trajectory and position in the next frame using the Kalman filter, then it tracks its bounding boxes across multiple frames with unique object IDs through the Hungarian algorithm, and eliminates bounding boxes for objects that have gone out of frame. DeepSORT adds deep learning features because it allows tracking to be additionally based on object appearance, compared to only relying on velocity predictions. This inclusion remediates the problem of hidden objects being erroneously marked as a different object when they reappear in later frames. The algorithm stages, along with its integration with a YOLO model, is shown in Figure 3 [12].

3.2 DeepSORT

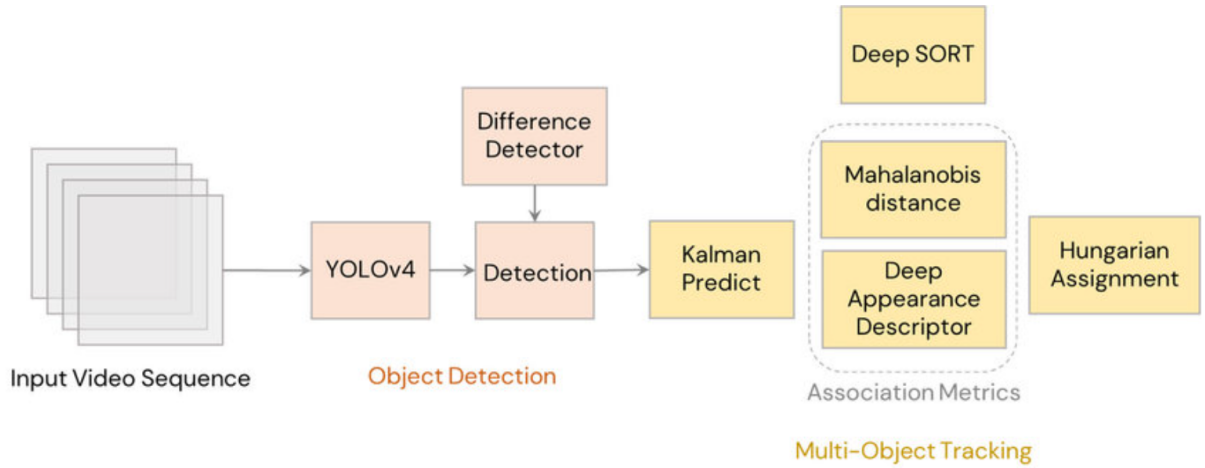


Figure 3: Main stage of DeepSORT algorithm

The final topmost layer of our system will perform further logic for analyzing the states of the lights. The primary additional state we aim to detect is flashing yellow, as this is commonly found on roadways. We plan on making use of DeepSORT output information like object ID, number of frames since last update, bounding box dimensions. The number of frames since the last update is an important metric associated with any detected object and is recorded by DeepSORT in its Tracker class [13]. It is related to when an object gets hidden by the elements in its vicinity, as discussed earlier when talking about DeepSort deep learning, so we can extend this functionality to apply to scenarios when the traffic light stops for a few frames, only to reappear again due to blinking behavior. The stages of our project's algorithm are depicted in Figure 4.

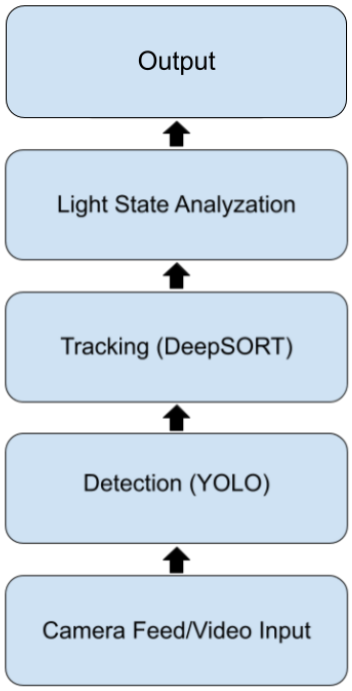


Figure 4: Stages of the project's detection algorithm

4.3 Approach for design validation (1 page, 5 points)

Our system needs to be able to perform a few tasks. The first and one of the most foundational is detection. We must be able to detect traffic lights quickly while also maintaining a high level of confidence and accuracy. We will seek a balance between model size, speed, and accuracy in our testing by training various model sizes, tuning parameters and comparing performance results. We will measure metrics such as frames per second and confidence of detections. In doing these model evaluations, it may be helpful to come up with some sort of aggregate measurement that combines many metrics in order to easily differentiate between model performance, so we seek to develop an aggregate performance measurement when we have more experience with the available and relevant metrics related to detection.

Further, our system must track multi traffic lights simultaneously across multiple frames. We will want to verify that this is being done accurately and with minimal ID switches/loss of tracks. It will also be helpful to ensure that our tracking works reliably with object occlusion and possible dropped detection frames. At the topmost layer, we will want to make sure that our further logic for state detection i.e. flashing yellow lights is able to work reliably.

Lastly, our ROS pipeline should function as needed in line with the project requirements. Our nodes should be receiving the correct inputs and outputs in the correct formats and we should be outputting data in the specifications provided by the client. Our model should also be a specific output, which is ONNX format.

5 Engineering standards (25 points)

5.1 Project management (1 page, 10 pts)

Aaryan Shenoy has worked as a software developer intern for the Huntsman Corporation and General Motors, where he has gained experience in project management and system engineering. He has also taken coursework in Distributed Systems and Database Systems. Due to this software engineering experience, he will have the role of system design. Aaryan will propose ideas and implement strategies related to the architecture design of the ROS and machine learning framework.

Morgan Roberts worked as an intern at Upshur Rural Electric Cooperative and led several collaborative coding projects in the IT department. Morgan aspires to take the leadership and collaborative experience gained from these opportunities and utilize them as a member of the team. As a team leader, Morgan aims to help with all parts of the project.

Xiaohu Huang worked as an intern at FlightSafety International and has experience with satellite image processing. He also has some knowledge of computer vision and machine learning from his coursework. Due to these experiences, he will have the responsibility of software design.

Clayton Gowan has worked at Capital Technology Group as a software development intern. He also has some high level knowledge of computer vision from working with the TAMU RoboMaster Robotics team. Due to his past software development experience and familiarity with computer vision concepts, he will be assigned the responsibility of software design.

Robert Madriaga has a relevant academic background with machine learning and artificial intelligence from course experience at Texas A&M University, which they can apply to this project. They also have experience working with robotics from the TAMU RoboMaster team. For these reasons they will be in charge of technical writing and training the model.

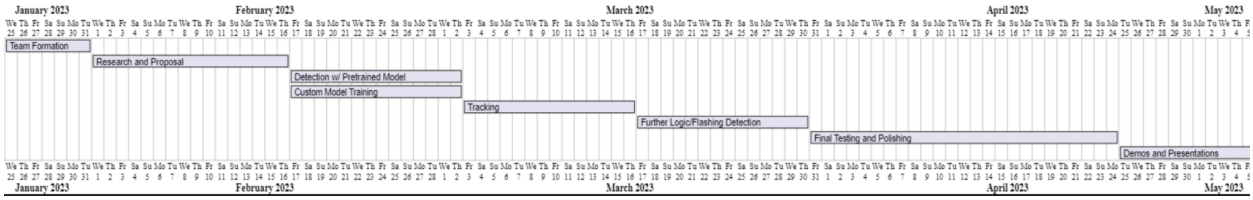
Technical reporting will be divided evenly among team members. This should allow each member to grow familiar with the required level of reporting for projects further down the road. Each week, a different team member will be responsible for the weekly report. All members should have access to the document and be involved in the revision process prior to submission.

Besides weekly meetings with the professor and TA, the team will meet outside of class to collaborate and discuss the project on a regular basis. Each member of the team will be responsible for stating what they accomplished with respect to what they have been assigned. The team will coordinate and hold each other accountable by providing updates to a common todo list. The team will also use a Gantt chart to aid in the visualization of overall progress and stages towards completion of the project.

5.2 Schedule of tasks, Pert and Gantt charts (1 page, 5 pts)

Break down the project into clearly identified sub-tasks, analyze dependencies among them, identify critical paths, and design a feasible schedule for accomplishing these tasks.

The project has been divided into different components and stages in the gantt chart shown below. This schedule, while subject to change, will serve as a general guideline for the team’s progress goals.



The first stage of development will be split into two parts. A portion of the team will focus on getting our basic ROS environment/pipeline setup and working with YOLO detection on some pre-trained models/weights in order to verify that the system works correctly. At the same time, another portion of the team will begin work on training different custom models so that when detection is ready to work with ROS we will be able to start testing. Second, the tracking portion of the system will be developed using output from the detection system. The detection portion is a major dependency for tracking to function as intended and will be needed to be completed first. The last development stage will be adding any further logic such as flashing yellow detection. Following development, there is a period reserved for testing and polishing which can also serve as a time to fix or work on any areas still in need of improvement.

The primary critical path for development includes getting ROS setup, getting detection working correctly, and making sure we are working with the correct format for inputs and outputs. The reason that this path is critical is because the project cannot function or grow without the ROS foundation or detection and cannot be integrated into the overall system without correct formatting, inputs and outputs. The period to complete the project development is from roughly February 17 to April 24 2023.

Within each primary stage of the project, the team will assign tasks according to team roles and hold meetings in order to work together and keep track of development. The gantt chart will be referenced along the way as a guideline for progress goals and will be adjusted as needed to account for any changes in requirements or development scheduling.

5.3 Economic analysis (1/2 page; 3 pts)

Autonomous vehicles in general have been a highly controversial and popular topic within the last few years and have been growing in their abilities with wider adoption in the market. These systems are highly marketable and could feasibly make up a significant portion of the automobile industry in the near future especially with the push from major manufacturers and governments to move to electric vehicles.

Sustainability:

Our project will be using well known open source software for detection, classification, and tracking and running on ROS2, so any vendors can be used as long as the hardware can run Unix and are compatible and powerful enough to run the system. As time goes on the software will need to be trained on new data to accommodate changes to traffic light infrastructure or completely new data sets to be compatible with traffic lights from multiple countries as the adoption of autonomous vehicles grows. New regulations regarding autonomous vehicles will also require updates to the software to ensure compliance with changing laws.

Manufacturing:

Even though our software is compatible with any Unix operating system that can run ROS, improving hardware performance can have unexpected consequences. Such as how our system will work when dealing with higher resolution and frame rate cameras and changes in computer architecture change the softwares behavior. Since we are using open source software, lack of updates or new implementations can cause version compatibility issues when having to work along with more modern and updated software in the autonomous system.

5.4 Societal, safety and environmental analysis (1/2 page; 2 points)

Autonomous vehicles have the potential for great societal benefits in the way of convenience and quality of life. If implemented in a widespread, robust and correct manner, they could result in more optimal travel times and offer increased accessibility to transportation. On the other hand, there are concerns about privacy due to the monitoring equipment required to make the vehicles function.

There are also numerous safety concerns when it comes to autonomous vehicles. These systems must be designed with extreme precaution, as any error in the system could result in the loss of human life and/or damage to property. Simple errors on both the software and hardware sides could easily be catastrophic. However, autonomous vehicles also have the potential to offer many safety benefits. A wide or full 360 degree field of vision working in conjunction with fast computational algorithms theoretically gives a system that is operating under safe conditions the ability to detect and respond to situations faster than humans can and with more consistency.

Concerning the environment, optimal autonomous vehicles could potentially reduce emissions by creating shorter travel times overall. Additionally, most of these vehicles are electric by nature. On the contrary, an inefficient navigation algorithm could result in more pollution/emissions due to longer travel times.

5.5 Itemized budget (1 page; 5 pts)

Given that this is a software-based project which will utilize free and open source tools, we do not expect to incur any expenses.

6 References (4 points)

[1] Q. Wang, Q. Zhang, X. Liang, Y. Wang, C. Zhou, and V. I. Mikulovich, “Traffic Lights Detection and Recognition Method Based on the Improved YOLOv4 Algorithm,” *Sensors*, vol. 22, no. 1, p. 200, Dec. 2021, doi: 10.3390/s22010200.

[2] M. Mostafa and M. Ghantous, "A YOLO Based Approach for Traffic Light Recognition for ADAS Systems," 2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 2022, pp. 225-229, doi: 10.1109/MIUCC55081.2022.9781682.

[3] K. Alkiek, “Traffic light recognition - A visual guide,” *Medium*, 02-Oct-2018. [Online]. Available: https://medium.com/@kenan.r.alkiek/https-medium-com-kenan-r-alkiek-traffic-light-recognition-505d6ab913b1. [Accessed: 16-Feb-2023].

[4] M. Bach, D. Stumper and K. Dietmayer, "Deep Convolutional Traffic Light Recognition for Automated Driving," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 851-858, doi: 10.1109/ITSC.2018.8569522.

[5] K. Behrendt, L. Novak and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 1370-1377, doi: 10.1109/ICRA.2017.7989163.

[6] S. R. Maiya, “DeepSORT: Deep learning to track custom objects in a video,” Nanonets AI & Machine Learning Blog, 24-Apr-2020. [Online]. Available: https://nanonets.com/blog/object-tracking-deepsort/. [Accessed: 10-Feb-2023].

[7] Q. -C. Mao, H. -M. Sun, Y. -B. Liu and R. -S. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," in IEEE Access, vol. 7, pp. 133529-133538, 2019, doi: 10.1109/ACCESS.2019.2941547.

[8] M. Bach, D. Stumper and K. Dietmayer, "Deep Convolutional Traffic Light Recognition for Automated Driving," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 851-858, doi: 10.1109/ITSC.2018.8569522.

[9] A. Kingery. CSCE 482. Class Lecture, Topic: “The Robotic Operating System.” College of Engineering, Texas A&M University, College Station, TX, Jan. 26, 2023.

[10] M. B. Jensen, “Lisa Traffic Light Dataset,” *Kaggle*, 28-Feb-2018. [Online]. Available: https://www.kaggle.com/datasets/mbornoe/lisa-traffic-light-dataset. [Accessed: 16-Feb-2023].

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] Sanyam, “Understanding multiple object tracking using DeepSORT,” LearnOpenCV, 11-Nov-2022.[Online].Available:https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort. [Accessed: 10-Feb-2023].

[13] “Yolo v3 real-time object tracking with deep sort,” *PyLessons*. [Online]. Available: https://pylessons.com/YOLOv3-TF2-DeepSort. [Accessed: 16-Feb-2023].

7 Appendices (1 point)

7.1 Product datasheets

Include product datasheets that may be particularly relevant to your proposed work. Say you want to use a certain type of microcontroller, because it has just the right combination features (e.g., types of I/O ports, or power consumption, etc). You would then attach datasheets for this product.

NOTE: including a data sheet does not replace the need for explaining how the component works and how it will be integrated in the system (section 4.2).

7.2 Bios and CVs

Morgan Roberts: I am a senior computer science major at Texas A&M and I am minoring in cybersecurity. I worked as an intern at Upshur Rural Electric Cooperative in the IT department and gained practical experience and knowhow about cyber threat mitigation.

Aaryan Shenoy: I am a senior computer science major at Texas A&M. I have gained software engineering experience through my internships at Huntsman Corporation and General Motors. In these companies, I worked on building applications for automating manufacturing processes and optimizing encryption software. I was also employed as a Calculus II and III tutor for 2 years at the Texas A&M Math Learning Center.

Clayton Gowan: I am a senior computer science major at Texas A&M minoring in mathematics. I have previously interned at Capital Technology Group as a software developer. I have some familiarity with computer vision concepts from being a member of the TAMU RoboMaster Robotics computer vision team for three years. I am looking forward to learning more about computer vision and ROS through working with the team on this project.

Robert Madriaga: Howdy, I’m Robert Madriaga. I am a senior majoring in computer science with a minor in mathematics at Texas A&M. I have experience with robotics as a member of the TAMU RoboMaster Robotics embedded systems team for three years. I am excited to learn more about autonomous vehicles and computer vision technology by working on this project.

Xiaohu Huang: I am a senior computer science student at Texas A&M minoring in cybersecurity and statistics. Previously, I have interned at FlightSafety International, an aerospace company specializing in full motion realistic flight simulation for commercial and military uses. During my internship with FlightSafety, I gained experience with satellite image processing, using an industry standard IDE(visual

studio) and working in a team based environment. Through this project I hope to gain experience in computer vision and technologies used in autonomous vehicles.

Morgan Roberts

Computer Science Student at Texas A&M University

907 Cross St.
College Station, TX 77840
(903) 522-0927
morgan.roberts00@yahoo.com
github.com/MorganRoberts00

EXPERIENCE

Mathnasium, Longview, TX — *Instructor*

August 2018 - July 2019

McAlister's Deli, Longview, TX — *Kitchen Staff*

May 2021 - August 2021

Upshur Rural Electric Coop, Longview, TX — *IT Intern*

June 2022 - August 2022

EDUCATION

Texas A&M University, College Station, TX — *Bachelor's in Computer Science*

August 2019 - Present

Cumulative GPA: **3.8/4.0**

Planned: Computer Science Major with Cybersecurity & Math Minors.

Longview High School, Longview, TX — *High School & International Baccalaureate Diploma*

August 2015 - May 2019

PROJECTS

Rasterizer — *C++*

An implementation that allows object coordinate data to be encoded into an image file. Written without access to the OpenGL graphics standard.

Personal Website — *HTML, CSS, JS*

A website written from scratch that includes details about my personal life and hobbies.

My Concerts — *Django, Bootstrap*

A website that allows users to find concerts in their local area. This was a team project and I primarily worked on front end development.

SKILLS

Programming Languages

- C++
- Python
- Java
- JavaScript
- PostgreSQL
- Haskell
- PowerShell

Familiarity with object-oriented and functional programming paradigms

Experience with Collaboration and Problem Solving in a Team Setting

AWARDS

Highschool National Honor Society member.

Highschool A-Honor Roll.

Graduated highschool in the top 2% of class with an International Baccalaureate Diploma.

LANGUAGES

Native English

Basic Spanish

Basic Japanese

Aaryan Shenoy

6 Pirouette Place, The Woodlands, TX 77382
(408) 306-2767
shenoy1011@gmail.com

SKILLS

Languages and Tools

- C++, Java, SQL, Git

Soft Skills

- Communication, High Learning Curves, Taking Initiative

EXPERIENCE

TEXAS A&M UNIVERSITY — **College Station, TX**
Math Learning Center Tutor — *January 2021 – January 2023*

- Provided coursework help for students in Calculus II and III
- Taught remotely and in-person to students in one-on-one and group settings

GENERAL MOTORS — **Austin, TX**
Software Engineer Intern — *May 2022 – August 2022*

- Completed feature implementations to overhaul encryption infrastructure in vehicle units
- Developed microservices using Java, PostgreSQL, Spring Boot, Git, and REST APIs
- Worked in a team that utilized agile methodologies and test-driven development
- Collaborated with 8 team members in code reviews and application architecture designs

HUNTSMAN CORPORATION — **The Woodlands, TX**
Automation Analyst Intern — *June 2021 – August 2021*

- Delivered a logbook and action tracking app for plant operations
- Implemented frontend in PowerApps
- Consulted with third-party sources on backend connections with Microsoft SQL Server
- Facilitated project handover with videos and requirements documentation

TEXAS A&M UNIVERSITY — **College Station, TX**
Research Assistant — *August 2020 – May 2021*

- Improved program code for data analysis of cattle social behavior
- Obtained experience in R

EDUCATION

TEXAS A&M UNIVERSITY — **College Station, TX**
Bachelor of Science in Computer Science — *August 2019 – May 2023*
GPA: 3.8/4.0
Courses: Data Structures & Algorithms, Distributed Systems, Database Systems

Phone: (254) 718-4949
E-mail: cgowan@tamu.edu

Clayton Gowan

https://github.com/CGowan44
www.linkedin.com/in/clayton-gowan

EDUCATION

Texas A&M University — **College Station, TX** — **May 2023**

- Bachelor of Science in Computer Science; GPA: 4.0

Temple High School — **Temple, TX** — **June 2019**

- Rank 1/479; Valedictorian
- Graduated with International Baccalaureate Diploma

LANGUAGES, TECHNOLOGIES, AND SKILLS

- C++, Python, JavaScript, HTML/CSS, EJS, Haskell, Java, Dart, HDL, C#, SQL
- Git, Node.js, MongoDB, Vue.js, React.js, Flutter, VSCode, Unity, Visual Studio
- Interests: Web & Mobile Development, Computer Vision, Security, Game Development

EMPLOYMENT

Software Development Intern — **Capital Technology Group** — **Summer 2022**

- Worked in collaboration with U.S. Digital Response to further the development of an enhanced government grant identification system.

TECHNICAL PROJECTS & INVOLVEMENT

- **TAMU Robomaster Robotics Computer Vision Team (Fall 2019 - Spring 2022) -** Utilized Python and OpenCV to develop a system to detect, track, and predict the movements of enemy robots. Served as team lead for spring 2022.
- **Daily Log Website (Summer 2021) -** Created and deployed a website that uses a MongoDB Atlas database to store individual user posts with login and CRUD functionality.
- **Weather App (Summer 2020)** - Created a weather app for Android using Flutter and Dart along with the IQAir AirVisual API to obtain air quality and weather data for the current location.
- **Sudoku Solver (Summer 2020)** - Created a sudoku solver app for Android using Flutter and Dart, stemming from earlier versions in C++ and Python.
- **Unity Game (Summer 2019)** - Created a children's game in Unity using C# designed to be used for the learning of basic colors, shapes, and letters.

COMMUNITY SERVICE

Volunteer — **Ronald McDonald House Charities of Temple, TX** — **Summer 2021**

- Performed various duties including processing aluminum cans, installing ceiling fans, and cleaning throughout the summer as needed.

IB CAS Project — **Temple, Texas** — **Mar 2018 – Sep 2018**

- Planned and organized the cleanup of Hodge Cemetery in Temple, TX with two other students; Involved communicating with staff and clearing brush/trash; Worked in collaboration with the Bell County Work Release Program.

HONORS & ACHIEVEMENTS

- **CompTIA ITF+ Certification (Summer 2019)** - ITF+ Certified as of 07/23/2019.
- **National German Exam Gold and Silver Medal Recipient (Jan 2017 - Jan 2019) -** Achieved 1x Gold and 2x Silver medals.
- **Bell County Sheriff's Foundation Scholarship (2019 - 2022)**
- **VeraBank Scholarship (2019, 2021)**

Xiaohu Huang

Creative, motivated & tech savvy student, US Citizen
Seeking internships and co-op positions

huangxia000@tamu.edu (preferred)
linkedin.com/in/maxhuang000
(713) 328-9969
College Station, TX, 77840

Education

Texas A&M University 2019 - Fall 2023 **B.S. Computer Science** Dean's Honor Roll

- Major GPA 4.0 Overall GPA 3.5 Minor in Statistics, Cybersecurity Expected Graduation Fall 2023

Coursework

- **Data Structure & Algorithms :** Object Oriented Programming in C++, Time complexity, Graphs, Hashing
- **Intro to Machine Learning :** Parametric and non-parametric methods (KNN) in Python
- **Principles of Statistics II :** Statistical Analysis of Data Sets using R

Employment

FlightSafety International Summer 2022 **Database Tools Intern**

- Developed and improved existing database & visualization tools for use by our modelling and imagery team
- Worked with power of 2 mapped satellite imagery, worked on 20+ year old legacy codebase
- Gained experience in Perl, C++, Visual Studio

Skills & Qualifications

- **Development:** Proficient : **C++** Familiar : **Java, Python, R, HTML, CSS**
- **Software Skills :** Proficient : **Adobe Creative Suite, Autodesk AutoCAD** Familiar : **GitHub, R studio, Jupyter Notebook**
- **Language Skills :** Native: **Chinese - Mandarin** Familiar : **French, Japanese**
- **Other Skills :** Woodworking, Architectural Design, Graphics Design, Photography

Projects

KaraEcho Alexa Skill devpost.com/software/karaecho-karaoke-amazon-alexa-skill

- Alexa Karaoke Skill, removes vocal from any song on demand using machine learning
- Gcloud Flask server retrieves & processes song from streaming service (YouTube), then send to Alexa for playback
- Built with AWS Lambda, Gcloud, Flask

Seam Carving github.com/MikeAlphaXray/Seam-Carving-cpp

- Content Aware image scaling, scales image while preserving proportions of the subject
- Based on the paper presented at ACM SIGGRAPH 2007

Awards

ACC General Motors Personal Website Contest - 3rd Place mikealphaxray.github.io

- Built Fully Responsive Personal Website with HTML + CSS

Div Hack 2021 - Most Creative Award

- Built Augmented Reality app with SparkAR, App allows user to view 3d model in AR space on a mobile device

Extracurriculars

Clubs & Organizations

- **Aggie Coding Club** - Member Currently pursuing officer position
- **Aggie Web Developers** - Member Attended workshops, Intro to Version Control with GitHub
- **Society of Asian Scientists and Engineers** - Member Attended workshops & meetings

Hackathons

- **HowdyHack 2021** - 24 hour hackathon in team of 4, developed KaraEcho Alexa Skill, gained experience in AWS Lambda
- **Div Hack 2021** - 24 hour hackathon in team of 2, developed SparkAR Augmented Reality app