SEMANTIC ROAD DETECTION FOR NON-IDEAL RURAL ROADWAYS WITH LIMITED APRIORI MAPS

A Thesis

by

JONAS LOSSNER

Submitted to the Office of Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2023

Major Subject: Engineering Technology (Mechatronics)

ABSTRACT

In this work we provide a system that has the ability to detect the entire road area ahead and provide semantic information about where to move within that area for an autonomous vehicle on a wide array of road types in rural environments. This approach combines current state of the art image road segmentation, image lane detection and lidar ground segmentation techniques by fusing the output of algorithms based on these techniques to produce a costmap to be used by a local planner for motion planning and vehicle control.

One of the main goals of this thesis was to prove whether this is a sensible task given the current state of the art in the relevant algorithms and computational resources, and to propose improvements that address the limitations of this work. These limitations were determined by applying the proposed system to the real-world driving scenarios in simulation and by evaluation on a test vehicle driven on real public rural roadways.

We have presented a functional system that works within the assumptions and limitations set, and shows higher accuracy results compared to the traditional approach.

DEDICATION

I dedicate this work to the memory of Mr. James K. B. Nelson, whose generosity and wisdom continue to inspire me.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Langari, and my committee members, Dr. Gopalswamy, Dr. Song for their support throughout the course of this research and prior to this research.

Finally, thanks to my family, Aggie family and friends who have supported me throughout my time at Texas A&M University.

CONTRIBUTORS AND FUNDING SOURCES

No outside contributors collaborated on this work. Graduate study was supported by a fellowship/stipend from Texas A&M University. This work was also made possible in part by USDOT/TxDOT under USDOT Project Number 693JJ319NF00001 and TxDOT Project Number 0-7099. Its contents are solely the responsibility of the author and do not represent the official views of the UDOT/TxDOT.

NOMENCLATURE

Algorithm	a program that solves a specific task
CVAT	Computer Vision Annotation Tool
HD Map	High-Definition Map
ILD	Image Lane Detection
IRS	Image Road Detection
LGS	Lidar Ground Segmentation
Mapless	Navigation without the use of HD maps
Maplight	Navigation with minimal roadway maps
MC	Motion Control
Model	machine learning model, defines a neural network to perform a task
MP	Motion Planner
OSM	Open Street Maps, Road Network Map
PSPNet	Pyramid Scene Parsing Network
RANSAC	Random Sample Consensus
RELLIS Campus	Texas A&M Testing Facility
RNP	Road Network Map

ROS	Robot Operating System, Open-Source Middleware
YOLO	“You Only Look Once”, Image Object Detection Algorithm
TP	True positive
FP	False positive
FN	False negative
Precision	purity of positive detections relative to the ground truth.
Recall	completeness of positive predictions relative to the ground truth
F1	relation between Precision and Recall
IoU	Intersection over Union
m.IoU	mean IoU
Acc	Accuracy, percent of pixels in the image which were correctly classified
aAcc	average Accuracy
mAcc	mean Accuracy

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

INTRODUCTION

One of the most important challenges for autonomous vehicles is to find a traversable path in whichever environment they may operate. This challenge is usually addressed by providing the vehicle automation system with a High Definition (HD) map that includes information about the environment in which the vehicle operates. HD maps can include lane markings, speed limits, curbs and other important elements of the roadway that can assist the automated vehicle to localize, navigate and guide itself along the road. HD maps are generally complex, expensive to produce and must be regularly updated to reflect any changes in the environment since vehicles using these maps rely heavily on the accuracy of the information provided in the map.

The majority of autonomous (or high autonomy) vehicles at this time are operated on well-structured roads with sufficient lane markings, namely freeways and arterial roadways. Examples of this are Ford's Blue Cruise and GM's Super Cruise systems which can only operate on a limited range of pre-mapped roadways at this time. Tesla claims to operate autonomously in additional environments without being fully reliant on HD maps but appears to be using some form of map , information on this is sparse and it cannot be determined whether these maps are considered HD maps or not [1]. Maps used by Tesla provide information about stop signs, lane direction and approximate lane location, but may include more. Waymo, Motional and TuSimple, companies that have demonstrated driver-out autonomous driving, were only able to achieve this by keeping extremely detailed maps of the driving environment alongside remote operators who can intervene at any given moment. This approach is not feasible in areas with little traffic to keep HD maps up to date[1] or poor cell reception (or lack stable means of communication) to enable remote (human) operators to intervene. Other brands, such as Honda, Subaru, among others, offer Advanced Drivers Assistance Systems (ADAS) to keep the vehicle in lane but may not function effectively or disengage when the road curvature is narrow, or road markings are obstructed.

Given this heavy reliance on a priori knowledge (c.f. in terms of HD maps) and the current limitations of automated road detection, it is impossible to have the current generation of automated vehicles operate in rural areas without endangering road users. The present study attempts to address this important problem with a view to utilizing an efficient approach to automated road detection that can be combined with minimal digital maps such as Open Street Maps (OSM) toward facilitating automated driving on rural roadways. To this we started with a brief review of literature followed by a formal identification and definition of the problem.

[1] The reason traffic density is mentioned is to signify that keeping maps up to date is not feasible as many rural roads are not traveled enough to update a map at sufficient rates to account for any changes.

LITERATURE REVIEW

Literature in the area of Autonomous Vehicles (AVs) is plentiful. Many papers discuss perception and motion planning in urban, suburban or highway environments but fail to address the specific case of rural roads. Unfortunately, there is little published work that shows major progress in the area of mapless or maplight navigation. Mapless and maplight seemingly are used almost interchangeably, mapless refers to the navigation without the use of HD maps, -less meaning not High definition, giving no clear guideline of "Definition" of the map, maplight specifically means use of only Road Network maps such as Google Maps or Open Street Maps.

Reference [8] is among the few in this area based on work completed at MIT in 2018. Using Open Street Maps, Ort et. al, set out to create a system for mapless navigation (mapless refers to not having HD, just basic a Road Network Map (RNP) similar to Google Maps) in rural areas augmented with Lidar sensor data. The augmentation is in the form of an online lidar road detection algorithm. This system goes one step further and covers the local planning component of the Motion Planner (MP) as well as Motion Control (MC). In this initial work, the MIT team used a parametric approach to find the road area using a texture-based approach (smoothness/flatness of the road is usually significantly different from non-road areas) combined with a RANSAC algorithm to find the road boundaries and calculate the optimal trajectory. RANSAC, or random sampling consensus, is an algorithm that filters outliers by fitting all points to a plane equation and rejects points based on certain thresholds, in this case resulting in the lidar data to be filtered into road points that fit within the plane. This did not allow for discrimination between lanes within the road (if applicable) and may not be able to sufficiently differentiate between road/non-road on gravel roads where this difference in texture may not be apparent.

As a follow-up work, [9] built upon the knowledge gained from [8]. The road segmentation approach was changed to utilize model-based Lidar segmentation rather than a parametric approach. By a model, we mean a neural network that has been trained with a specific set of annotated data and thus has learned a given task and can perform inference on data that it has not been trained on. In this approach convolutional neural networks were trained to detect the road area based on lidar data. While there was significant improvement, the system struggled to correctly segment the road on up/down-hill roads and required a significant amount of annotated data for these edge cases. Some additional work in this study included creating HD maps and updating road network maps (RNP) using this method. Eventually, they downgraded their localization system, instead of using a high quality RTK supported INS system, as RTK may not always be available due to lack of communication means and cost associated with high quality GPS devices, they relied solely on lidar supported odometry (vehicle data such as steering angle and wheel speeds filtered with lidar odometry) which required some additional manipulation of the data to correctly localize within the Open Street Map (OSM) environment.

Reference [10] gives a good overview of navigation methods albeit for a smaller robot: Map based, Map-building and mapless. As we are planning on going mapless in the project that has supported this work, I focus only on this area. In their review, the authors of Reference [1] mainly focus on image/visual based methods. This focuses mainly on navigating in an open area with the main goal being to avoid objects through optical flow, appearance-based methods or direct object recognition combined with feature tracking. Although this does not directly apply to our proposed method, one of their conclusions is that there are few mapless navigation strategies that have been shown to be successful but vision-based approaches show a large potential that is currently still untapped.

Reference [11] proposes camera and Lidar fusion in order to create a semantic 3D Map (HD Map) - although they produce a High Definition Map as their output, and not directly for processing on the vehicle for online control decisions, in structured, urban environments, the method behind the system is quite similar to our proposed fusion method. While they only focus on semantic segmentation of the road, objects, sidewalks, light poles, etc. but not segmentation of the road area itself they were able to show very good results in creating meaningful 3D maps. One major drawback mentioned in this reference is that the quality of their 3D map decreases when focusing on minimizing runtime which is due to the process of creating the 3D map in the first place and the specific algorithms used in parsing the Lidar data. Compared to this - as our system will only use a basic Lidar processing algorithm and not create high-quality 3D maps we can provide quicker runtime.

Reference [12], as part of their lane detection system, sets forth the idea that lanes need to be predicted ahead of what is visible to the vision system. While valid, by using a simple Road Network Map, we would be able to extract information much further ahead than any prediction algorithm would be able to with similar if not better accuracy and as we will not be extracting lanes but a costmap this is much more straightforward. A costmap is a predefined area divided into smaller grids around the vehicle which are assigned a weight between 0 and 1, 0 signifying no cost to traverse, 1 showing that area is not traversable/has a very high cost.

PROBLEM IDENTIFICATION

Many rural roads are poorly marked, have minimal signature and are not well maintained. In general, we define roads into two different categories: Marked and Unmarked. These can be further divided into: Well Marked & Well Paved (WMWP), Poorly Marked & Poorly Paved (PMPP), Unmarked & Paved (UMP), Unmarked & Unpaved (UMUP). This thesis limits categorization to two categories, combining WMWP and PMPP, as well as combining UMP and UMUP. During the annotation process we noted that many of the WMWP and PMPP roads had similar structure and features that may not be easily distinguished by the neural networks we aimed to train. This is the same case for UMP and UMUP. Thus we decided to combine these together.

As noted above, given their location and limited traffic use - keeping HD maps of rural roads up to date is not feasible. Additionally, some of these roads are not paved - often constructed of gravel and are prone to corrugation - which can lead to changing road structure over time, especially during or after heavy rain. This means that a system needs to be developed that can detect the roadway ahead with minimal a priori knowledge and which is capable of not only detecting the traversable area but also is able to semantically determine where within that area is the most appropriate path to follow.

We have explored several different lane detection algorithms, many of which have been trained and evaluated on large datasets such as `CULANE` [15], `TuSimple` [16] or `Llamas` [14] - all of which encompass a large variety of roadways but in practice fail to detect lanes at a reasonable level of accuracy on our test routes, both in the well paved & well marked case as well as the unmarked & unpaved case.

`CULANE` is a traffic lane detection dataset created in 2018 that encompasses a wide range of roads in urban and suburban environments in Beijing. Their format is accepted by many algorithms to train lane detection models. We have used the dataset for initial training and evaluation, and have adopted their format for our own dataset. This dataset includes 133,235 images, just under 100,000 of them having annotations.

The `TuSimple` Dataset was created as part of the TuSimple Benchmark and challenge competition. It encompasses lane detections for highways with different traffic densities. This dataset has 10,000 images.

`Llamas,` the Labeled Lane Markers Dataset, was created by Bosch in 2019, and consists of 100,000 annotated images. This dataset was created as part of their benchmark.

All of the previously mentioned benchmarks and their rankings of different lane detection algorithms were used to decide which algorithm we would adopt for this work.

We have also explored several different road segmentation algorithms which have been trained and evaluated on the `KITTI` dataset [17] which while performing better than the aforementioned Lane Detection algorithms on a broader array of road types still struggled to perform accurately on our test routes.

The `KITTI` dataset [17] was created by the Karlsruhe Institute of Technology as part of their Vision Benchmark Suite for stereo, optical flow, visual odometry, 3D object detection and 3D tracking. It provides raw data, as well as annotated data to benchmark algorithms for the tasks mentioned prior. We used the dataset to train road segmentation algorithms and evaluated them on our own data.

In spite of these limitations, we believe that the current state-of-the-art Image Lane Detection (ILD) and Image Road Segmentation (IRS) algorithms can be fused to cover a broad array of roads without the need for HD maps. Specifically, we propose a model-switching multi-step fusion of ILD, IRS and Lidar Ground Segmentation (LGS) algorithms based on their respective confidence levels along with some limited a priori information through publicly accessible maps such as Open Street Maps (OSM). The idea is that an outside global path planner can use the basic map information to give our fusion system an initial path/direction as well as some information on potential intersections similar to how human drivers utilize a navigation system such as those commonly available in mobile apps or specialized GPS based devices. Furthermore, this approach provides a contingency base in case all sensors fail to detect the road with sufficient confidence, which can happen in mountainous environments where a significant part of the roadway may not be visible to the sensors - for example driving up a hill will obstruct the road on the downhill portion of the roadway. Additionally, this may allow for emergency pull-over in case the system fails to detect any road boundaries. Given that OSM does not supply sufficient information as to the width of the road or what may be to the side of the road, it may not be safe to use for a pull-over maneuver and thus may not work in many cases.

METHODS

We consider three main detection algorithms: Image Road Segmentation (IRS) through `mm-segmentation` [6], Image Lane Detection (ILD) through `LaneAtt`, [2] and [3], and Lidar Ground Segmentation (LGS) through `Segmenters_lib` which is referenced in [4],[5] and [13].

Image Lane Detection

For obvious reasons, a lane detection algorithm traditionally does not work when there are no lane markings as many approaches (parametric or model-based) specifically look for the lane marking itself (such as Hough-transform and Canny edge-detection [18]). `LaneAtt` is strictly speaking not a traditional ILD algorithm but, similarly to `Yolo` (You Only Look Once) [19], detects features in imagery by utilizing a single-stage model. During training, the model learns to combine local and global features which allows it to detect occluded lanes or lanes without proper lane markings. Through testing on our own data sets, we have seen that on well-marked & well paved roads, a `LaneAtt` model trained on the `CULANE` dataset performs close to correct with only few false-negatives or incomplete detections, meaning it detects a lane but much shorter than the actual lane in the image. We have also found that with the standard datasets it cannot detect any lanes on gravel roads. However, by adding several hundred frames of gravel road with our own annotations we were able to detect large portions of our test area at the RELLIS Campus that are unmarked & unpaved (gravel road). By adding custom frames of the gravel road, overall accuracy decreases on well-marked roads which makes the case for why one likely needs *separate models* that are specialized for the *different types* of roads. This is the essence of our approach in this work. In order to create a custom dataset, we utilized `CVAT`

(Computer Vision Annotation Tool) [22] and developed a Python script to convert the `CVAT` format into the `CULANE` format [15]. `CULANE` is a dataset that was created in 2018 and its image annotation format is easy to use and accepted by many lane detection algorithms for training purposes. Thus, we decided to use this format for our own annotations as they could be reused for future works on other algorithms.

The aforementioned approach shows that the LaneATT, lane detection model can be trained to detect lanes on the best-case and worst-case road types. We have also tested the cases in between when there are poor markings with good pavement or poor to no markings coupled with poor pavement and confirmed that we are able to train the algorithm using our own sufficiently annotated datasets to perform the task well. Figure 1 shows lane detection through `LaneAtt`. The blue lines represent the detected lane marking. While these are not absolutely correct they are close to the true lines. Additionally, it is important to highlight that in the center of the road only one line is detected - this is due to the fact that a double yellow line represents a centerline and as such is automatically combined into one line by the LaneAtt, lane detection model.

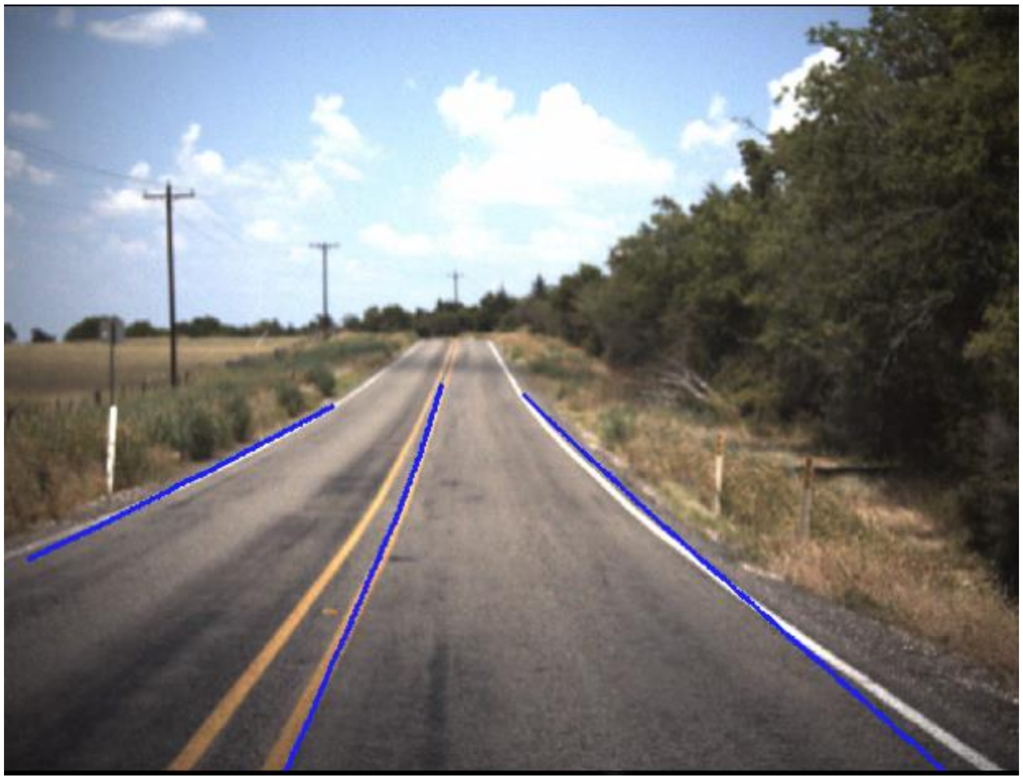


Figure 1: Lane Detection Output Example

Image Road Segmentation

Image Road Segmentation (IRS) works well in many scenarios and on many different road types but has one drawback as it does not classify the segmented road by any metric, meaning there is no discrimination between lanes, a driveway or emergency lane. This means that road segmentation identifies any area that can be considered road/traversable which can be useful for emergency maneuvers, such as swerving to avoid an obstacle. The point here is that this shows the need for using both lane detection and road segmentation in parallel to provide sufficient information to find the most appropriate area for the vehicle to move through.

In our application we use `mm_segmentations` implementation of the pyramid scene parsing network (PSPNet) [7] which uses a neural network to construct a feature map from which it uses the pyramid pool process to provide the final classified segmentation output through the convolutional layer. Similar to the previously mentioned issue of ILD not being able to detect the road area on poorly paved or unpaved roads using only one model we also tested the edge cases and found the same results again making the case for separate models that are specialized for different types of roads which we will discuss after explaining the IRS output.

Figure 2 shows a typical IRS output. As can be seen, the road area is colored deep pink. Obstacles, such as cars, will result in the areas not to be detected as road and are automatically removed as extraneous artifacts by the algorithm and thus are not shown in the figure. To highlight the difference from the output in Figure 1 it is evident that IRS goes beyond the actual road marking and shows the entire (in this case paved) area beyond the lane markings.

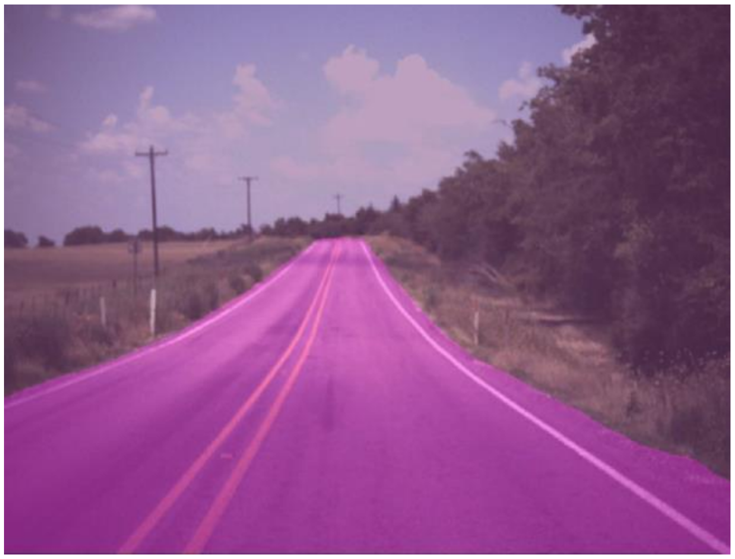


Figure 2: Road Segmentation Example Output

Switching Models

Since we want our system to cover as many road types as possible, we propose to use different neural network models for both the ILD and IRS cases which have been specifically trained for different road types. Specifically, when the system detects that the algorithm has a low confidence level on its detection for longer periods of time it will switch through the available models until the confidence is restored to a preset minimum value/threshold. The confidence values are obtained from each model along with the detected lane points and segmented road pixels. The original code for each algorithm has been modified in order to output these values. By a model, we mean a neural network that has been trained with a specific set of annotated data and thus has learned a given task (lane detection or road segmentation) and can perform inference on imagery that it has not been trained on. In this implementation we have chosen to average the confidence level over 10 detections (moving average) and reset the confidence value after every switch. This results in each model running for approximately 1 second before a model switch is performed. This can be easily modified but averaging 10 detections has shown the best performance.

Lidar Ground Segmentation

LGS, as the label suggests, is a method capable of dividing the Lidar Point Cloud (PCD) data into points that are i) ground, and ii) points above the ground that could be curbs, obstacles or other objects. Out of the box, LGS often only works in one environment as the configuration parameters need to be adjusted to certain environments. These parameters include allowable angle of the road, maximum distance that will be used from the vehicle outward as well as intensity values of the Lidar Point Cloud data. This is necessary as the ground structure on a gravel road vs a paved road is different in flatness and horizontal curvature.

Once a model has been determined to work, we are able to provide that information to the Lidar Ground Segmentation (LGS) algorithm - for example when the image lane detection and road detection switch to a model for gravel road we can pass parameters to the LGS algorithm to adjust to the environment.

Figure 3 depicts a visualization of Lidar ground and non-ground segmentation through `Segmenters_lib` which uses Ground Plane Fitting to detect ground-points and Region Euclidian Segmentation for non-ground points. `Segmenters_lib` also includes Ground RANSAC as an alternative for the ground segmentation which in our testing has performed poorly compared to Ground Plane Fitting, even after adjusting configuration parameters. Additionally, it also includes a non-region based Euclidean Cluster Segmentation for non-ground points, but this also performs poorly compared to the one we chose.

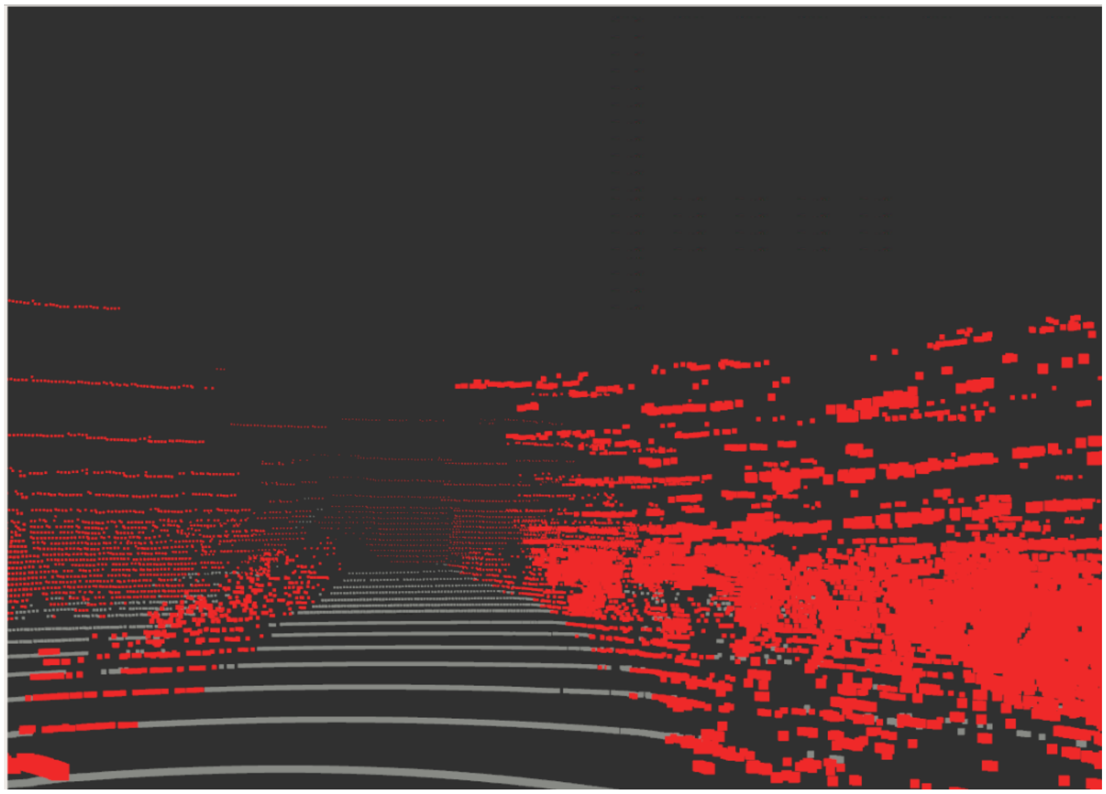


Figure 3: Ground/Non-Ground Segmentation Output Example

As previously mentioned, by using LGS, the original Lidar point cloud that includes all points is split into two different point clouds - visualized above, the gray points show ground points, red points show non-ground points.

Sensor Fusion/Projection

One Option for sensor/data fusion is at the raw level, meaning each image pixel is matched with a lidar point prior to be used as input to an algorithm [21]. Although this is a viable method, it requires very accurate extrinsic calibration (rotation matrix and translation vectors between each sensor). Thus, we chose to use fusion at a detection level, meaning the output of each algorithm is fused. The justification for fusing data at the detection level is that fusing data at the "raw" level requires very accurate calibration between all used sensors that must stay perfectly stable. Additionally, creating custom datasets for algorithms that do take fused data is more complex as well. Therefore, we will implement fusion at the "object/detection" level which is more robust to slight changes in calibrations. Moreover, as each algorithm operates by itself, when one algorithm fails to provide sufficient detections, we are able to continue operations based on the results from the alternate algorithm.

In order to project the 2D image detections into the 3D frame of the lidar sensor two equations are required to obtain the 3D points (from the Lidar Point Cloud) that correspond to each pixel in the camera image.

The calibration/parameters needed will be represented by

$$P_{cam_{lid}} = P_{cam} * R_{cam} * T_{cam_{lid}}$$

Where P_{cam} is the projection matrix, R_{cam} the rectification matrix of the camera (intrinsic calibration parameters) and $T_{cam_{lid}}$ represents the rotation matrix and translation vector between the camera and the Lidar sensor (extrinsic calibration parameters). These parameters can be obtained by using a relatively simple checkerboard calibration setup through ROS or MATLAB [20]. More advanced approaches utilize non-checkerboard targets, some use targetless calibration.

Once the calibration parameters are obtained the two equations used to obtain the Lidar points that correspond to the pixels are denoted in Figure 4 below [16]:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P}_{lidar}^{cam} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} \frac{u}{w} \\ \frac{v}{w} \end{bmatrix}$$

Figure 4: Lidar-Camera Projection Equations

Where [x_v, y_v, z_v] denote the 3D Lidar point coordinate and [u_c,v_c] denote the pixel corresponding to that point coordinate. The IRS and LGS algorithms output is transferred to the fusion system in the form of pixel locations. For IRS, we provide an interpolated line of [u,v] pixels. For the LGS, we provide the entire area of pixels classified as “road” in [u,v] pixels. Once each pixel is matched, we can extract and project the pixels of interest (detection outputs from ILD and IRS) to find their 3D location, most importantly, their depth as this is not possible based on monocular image detection alone. After the 3D location of each pixel is found, the costmaps can be computed in the next step,

Output

A costmap is a predefined area divided into smaller grids around the vehicle which are assigned a weight between 0 and 1, 0 signifying no cost to traverse, 1 showing that area is not traversable/has a very high cost. This is used for example to associate a higher cost with moving into a different lane or moving off the road where there are no obstacles in case of emergency. Additionally, a cost map can be easily added onto or be modified to fit other semantic mapping such as obstacles that are detected by one or more separate algorithms.

Other options such as potential fields or more vectorized outputs were considered but due to costmaps being used by many standard ROS local planners and obstacle detection algorithms this seemed most useful. Costmaps are also easy to visualize and thus preferred for presenting the idea at hand. The outputs can be modified to fit different systems, such as for local planners that do not use costmaps by adjusting the output format to the desired level, as aforementioned some local planners may use vectorized formats while others can utilize potential field maps or other means of communicating traversable area. To summarize, we implemented a road detection algorithm that outputs:

1. Any traversable area considered to be road defined by gravel or pavement
2. Any road area defined by lane markings or road edges
3. Cost of traversing different areas within 1. & 2.

by utilizing IRS, ILD and LGS.

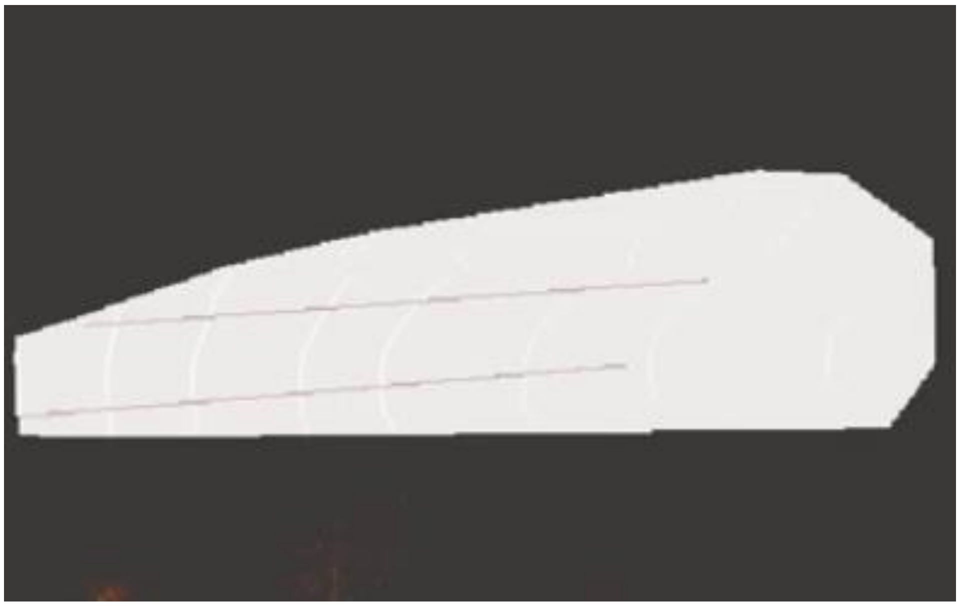


Figure 5: Costmap Output Example

Figure 5 shows a sample output of the generated costmap. The road area is shown as completely white area while everything surrounding is black. The road area starts out at an angle on the left side due to the overlap of the Field of View of the camera and the lidar, gradually growing wider to the right of the Figure. As the road area is the outline of all points that fall within the detected road area, the area may not be immediately recognizable as road to the human viewer. We chose to use 0 (minimum) cost for the detected road area while everything surrounding has a cost of 1 (maximum). The lane markings have a cost of 0.5, showing that it shouldn't be traversed through unless a different higher cost item such as an obstacle or end of road was to show up in the costmap. The lane markings cost of 0.5 was found to be reasonably high to prevent local planners from ignoring it but low enough to not impede the path if obstacle avoidance is required. The global costmap would be provided by a global path planner which is not covered in this thesis.

Image Detection Subsystem

Below Figure 6 depicts the proposed Image Detection Part of our system.

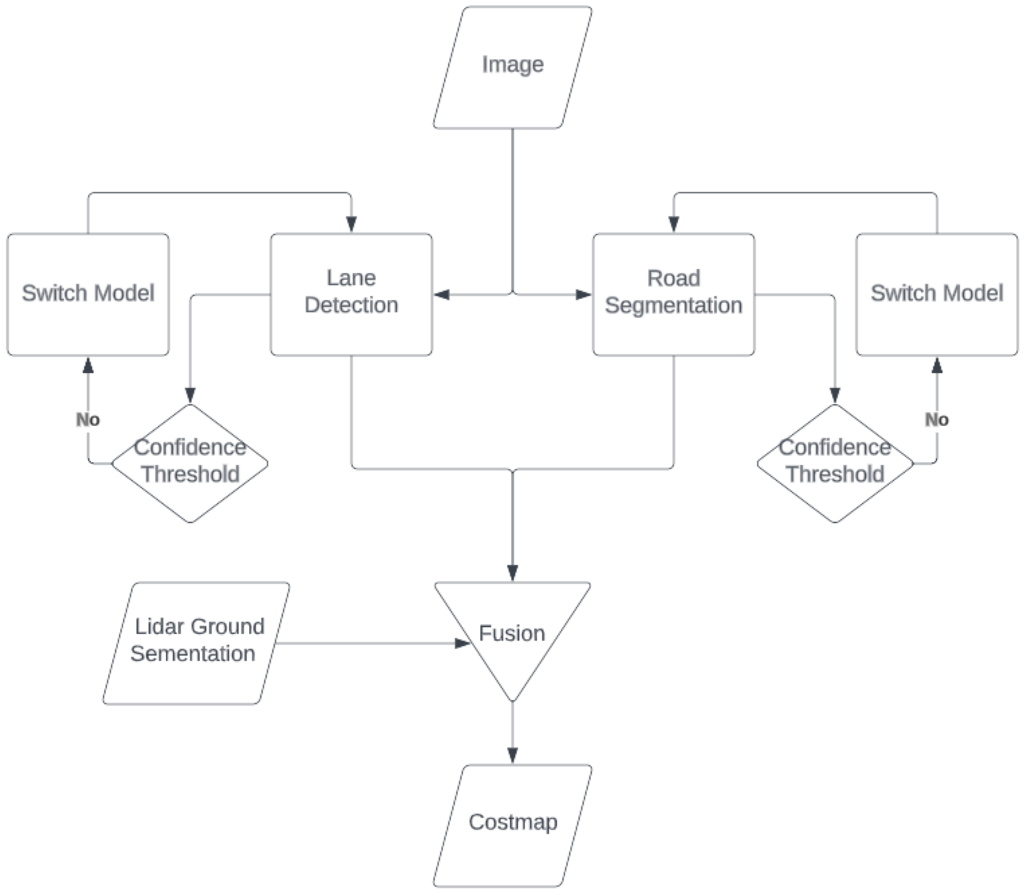


Figure 6: Image Detection Subsystem

The flowchart above shows the system of the Image Detection Algorithms. The image is received from the front camera and sent to both the ILD and IRS algorithms. To reiterate, both algorithms are able to switch to a different neural network model based on their confidence value. There are two models specifically trained for marked and/or unmarked roads. Finally, the two image detection outputs will be combined with the ground points from the LGS system through the previously described fusion process and converted into a costmap.

Lidar Detection Subsystem

Figure 7 depicts the Flowchart for the Lidar detection part of the proposed system.

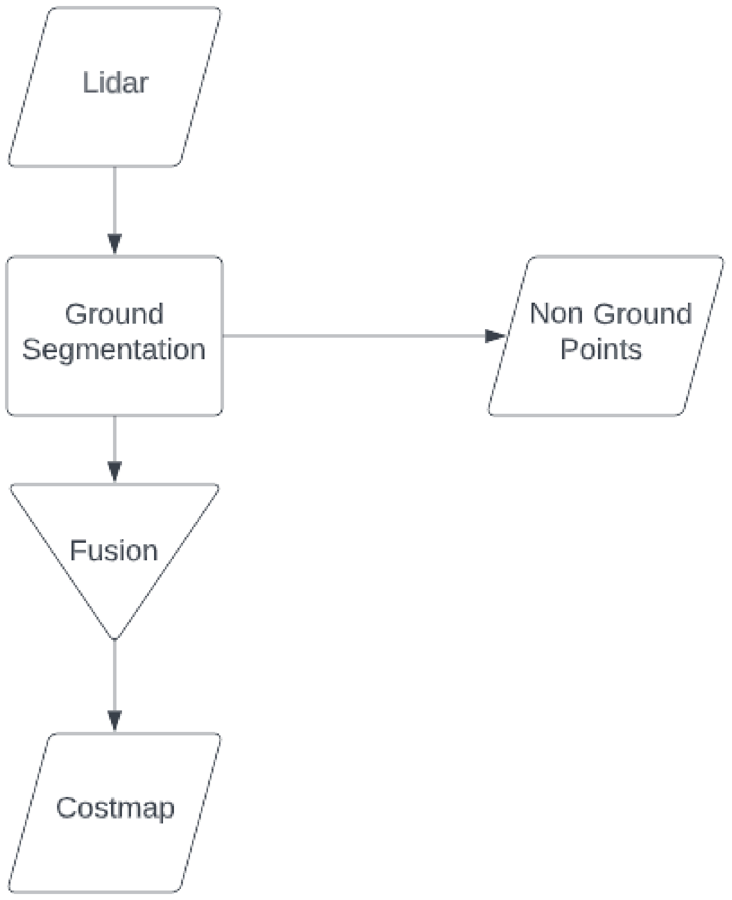


Figure 7: Lidar Detection Subsystem

The Lidar data is segmented into ground and non-ground points which can be used for object detection. The final ground output is additionally slightly filtered to minimize noise and points that are out of range to reduce computational overhead in the fusion step. Finally, as aforementioned, the ground points will be fused with the image detection results into the costmap.

ASSUMPTIONS

- All sensors are running at least at 10 Hz
    - Cameras can run around 30 Hz (depending on brightness etc.)
    - Lidar can run at 20 Hz
    - Global positioning through INS/GPS runs at 100 Hz with RTK support
- Computational resources on the vehicle can process detections in near real-time, or at least around the specified frequency of 10 Hz
- Extrinsic calibration between camera and lidar stay consistent with minimal change
- Cell reception is adequate to receive RTK GPS corrections

LIMITATIONS

- Vehicle travels at or below 55 mph (~24.6 m/s which at 10 Hz refresh rate will give detections every 2.46 meters)
- Sensors are not experiencing noise, e.g. due to inclement weather (snow, rain, fog or similar)
- Datasets used to train the neural networks may be limited due to time constraints and what data can be collected (for obvious reasons it would be very hard for us to collect data in the snow)

SIGNIFICANCE

Proving or disproving that “state of the art” detection algorithms can be leveraged in a non traditional way and fused to provide the roadway ahead of a vehicle in Real Time without detailed apriori maps. Additionally, specialized datasets for lane detection as well as road segmentation will be created and can be published along with the code implementation.

A different project - MapLite [8] and [9]- provides a similar problem statement: Navigation in rural environments in unmapped areas. While there are similarities in the implementation, MapLite relies solely on Lidar detection as well as Lidar segmentation and is not able to provide lane information. In my opinion, the advantage of using camera detection algorithms with switching between smaller, specially trained models, can reduce computational resources required as well as cost since cameras are cheaper than lidars. The lidar we use being a 32 plane lidar further reduces cost and resource requirements as the MapLite implementation requires a 64+ plane lidar for sufficient detections. Furthermore, many lidar detection algorithms have poor domain transfer, meaning that between different lidar sensors or environments the quality of the detections suffers.

TIMELINE

Standalone Image Detection Testing and migration to ROS1 - Spring 2022

Standalone Lidar Segmentation Testing and migration to ROS1 - Summer 2022

Data Collection and Annotation for creating our own trained models - Fall 2022 & Spring 2023

Fusion of all algorithms - Fall 2022

Testing, debugging and demonstration - Spring 2023 & Summer 2023

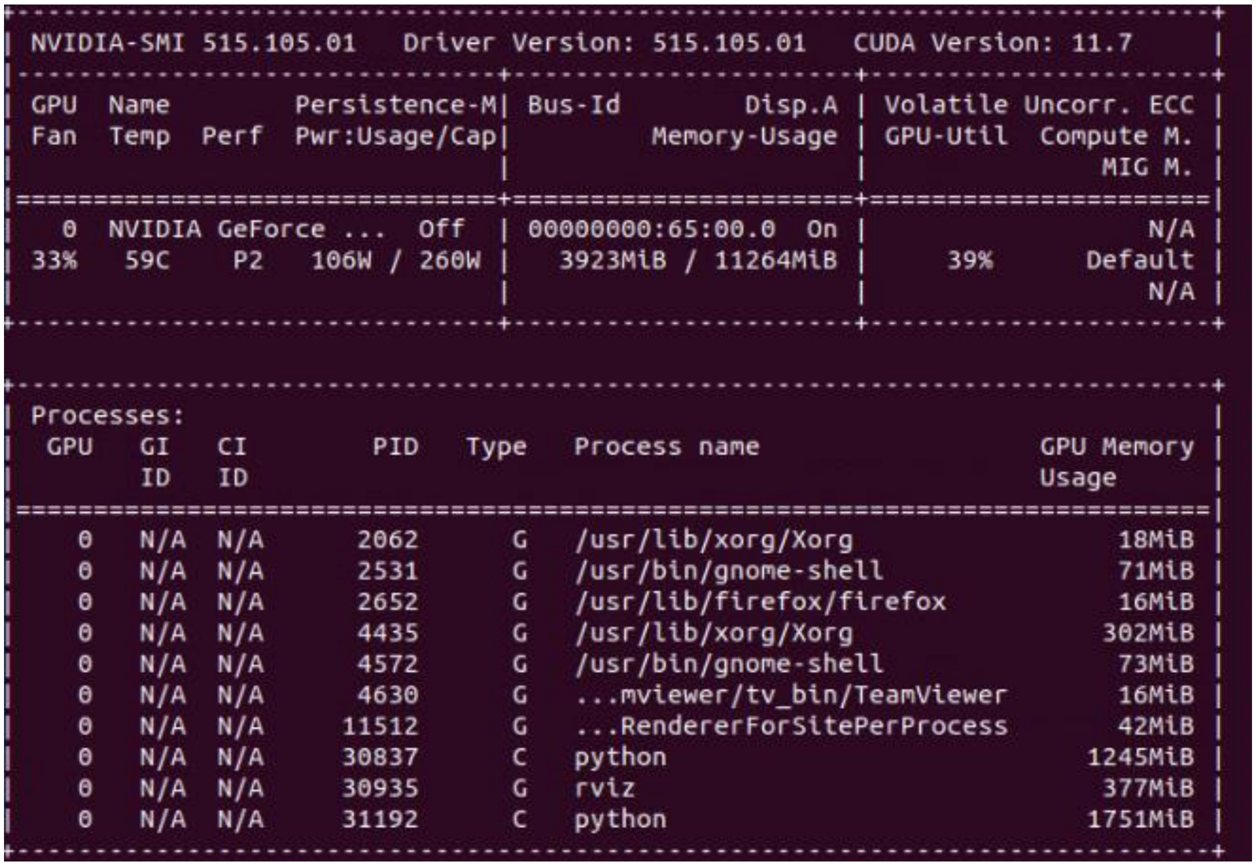
Final Analysis - Summer 2023

IMPLEMENTATION ANALYSIS

As proposed, the entire system is currently tested on Ubuntu 18.04 with ROS melodic. Some programs run in Python2 while others run in Python3 due to specific library dependencies. The system produces results at just around 10 Hz with room for improvement. The limiting factor is the time it takes for the models to infer as well as loading/unloading data from the GPU. This seems to be a CPU overutilization issue which could potentially be fixed by limiting the number of threads or using a different inference library - each which comes with its own limitations. Limiting the number of threads can further slow down inference data transfer to the GPU is limited. Using other inference libraries may require modifications to the actual detection models, simplifying the model and potentially reducing the detection accuracy.

The Lidar ground segmentation runs in C++ on the CPU, both image lane detection and image road segmentation run in python utilizing the onboard Nvidia GPU for inference.

The total available GPU memory is (11 GB) based on one NVIDIA GPU (RTX 2080 TI). A LaneAtt model requires 1200-1500 MiB. A `mm_segmentation` model requires 1500-1800 MiB. Additionally, the standard load is variable but when visualizing the data (images as well as Lidar point cloud) it varies between 750-1500 MiB. Figure 8 below shows the Nvidia system monitor (NVIDIA SMI) running the full system at just under 4000 MiB of GPU memory. This value fluctuates between 3600 MiB at the lowest, and 4300 MiB at the highest.



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.105.01   Driver Version: 515.105.01   CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:65:00.0  On |                  N/A |
| 33%   59C    P2   106W / 260W |   3923MiB / 11264MiB |     39%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      2062      G   /usr/lib/xorg/Xorg                 18MiB |
|    0   N/A  N/A      2531      G   /usr/bin/gnome-shell               71MiB |
|    0   N/A  N/A      2652      G   /usr/lib/firefox/firefox           16MiB |
|    0   N/A  N/A      4435      G   /usr/lib/xorg/Xorg                302MiB |
|    0   N/A  N/A      4572      G   /usr/bin/gnome-shell               73MiB |
|    0   N/A  N/A      4630      G   ...mviewer/tv_bin/TeamViewer       16MiB |
|    0   N/A  N/A     11512      G   ...RendererForSitePerProcess       42MiB |
|    0   N/A  N/A     30837      C   python                           1245MiB |
|    0   N/A  N/A     30935      G   rviz                              377MiB |
|    0   N/A  N/A     31192      C   python                           1751MiB |
+-----------------------------------------------------------------------------+
```

Figure 8: Nvidia system monitor Output

Switching between models does not take significant time or any additional resources on the GPU as the models are held in RAM memory until switched. The entire system consumes less than 5 GB of RAM memory; 1.9 GB for lane detection, 2 GB for road segmentation, 200 MB for Lidar Ground segmentation, 250 MB for fusion and 100 MB for costmap generation . The goal was to only utilize one GPU, as in the future, other algorithms such as Image Object Detection or Lidar Object Detection also need to be able to run simultaneously on a vehicle that has two GPUs, which presently require much more resources than the algorithms used in this thesis. With the system, as implemented, this is already possible but future work may free up more resources.

The annotations used were randomly chosen from several hours of recorded images and classified into unmarked (paved and gravel unmarked), as well as marked (paved 1 way and 2 way marked) for both Lane Detection and Road Segmentation. Annotations were completed through CVAT.AI and converted into the correct format needed for training for each LaneAtt and MMSegmentation. About half of the annotated data was used to train the model, while the rest of the annotations were used to validate the models and provide test metrics.

Annotation	**Road Type**	**# Images**
Lane Detection	Paved, unmarked	401
Lane Detection	Gravel, unmarked	290
Lane Detection	Paved 1 way, marked	300
Lane Detection	Paved 2 way, marked	240
Road Segmentation	Paved, unmarked	401
Road Segmentation	Gravel, unmarked	290
Road Segmentation	Paved 1 way, marked	300
Road Segmentation	Paved 2 way, marked	701
	Lane Total	1231
	Segmentation Total	1692

Table 1: Annotations

In total, annotation for lane detection resulted in 1231 individual images with annotations. Annotations for road segmentation resulted in 1692 individual images with annotations. Several lane detection annotations were thrown out due to no markings or discernable edges being present, which is why the number of lane detection annotations is lower than the number of road segmentation annotations. Overall, the selection, annotation and format transfer process took over 2 months to complete by myself. This presents a large bottleneck in training models, as training data that is available publicly is generally not classified by road type or environment in the way that was needed for this system.

Model	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1**
Unmarked	128	295	452	0.3026	0.2206	0.2552
Marked	77	153	178	0.3347	0.3019	0.3175
Combined	85	646	622	0.1162	0.1202	0.1182

Table 2: LaneAtt Training Metrics

Table 2 above shows the metrics provided by the training procedure for LaneAtt. The models shown, (Unmarked, Marked, Combined) have been trained, validated and tested on the annotations provided above, and no other outside data. With this, the Combined model performs significantly worse than the Unmarked/Marked models by themselves. As expected, the unmarked model performs slightly worse than the marked model, as it is not detecting perfect lines every time but rather the edge of a road which is not always very straight, especially on gravel roads. Overall, the Precision, Recall and F1 scores are not great for any of these models, which is likely the result of a limited dataset.

Model	aAcc	mIou	mAcc	Iou.road	Iou.bkgr	Acc.road	Acc.bkgr
Unmarked	0.9941	0.9869	0.9931	0.9912	0.9826	0.9962	0.9900
Marked	0.9541	0.9056	0.9410	0.9306	0.8806	0.9967	0.8853
Combined	0.8910	0.7964	0.8791	0.8791	0.7585	0.9946	0.7635

Table 3: MMSeg PSPNet Training Metrics

Similar to the LaneAtt results, Table 3 shows the training metrics for PSPNet based on the MMSegmentation training procedure. The models were trained with only two classes, being road and background (or not-road) denoted by .road and .bkgr in the table. Once again we see that the models that are split do perform better on their respective test annotations. In these metrics we see that the unmarked model actually performs the best, as I believe these types of roads have more distinct features or colors that may be easier to differentiate for the model. In general we can say, the combined model performs less accurately than the separated models on the marked/unmarked roads as shown by the training result metrics.

Below will be several images showing the result on other test data that was collected - that neither of the models had been trained on prior. It is important to note that there are various sections that look nothing like the training data used to train the models while others may look similar to the training data. Overall, the segmentation combined model behaves similar to the separated models in certain instances, for example when the road ahead looks similar to the training data. In those cases, the separated models show a slightly higher accuracy, as can be seen in Figure 9 and 10.

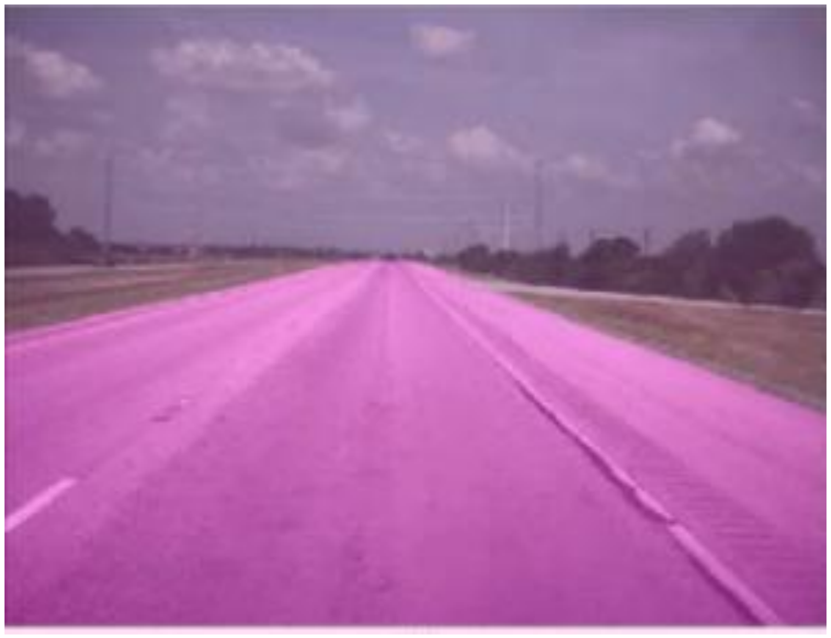


Figure 9: Separated Model, marked

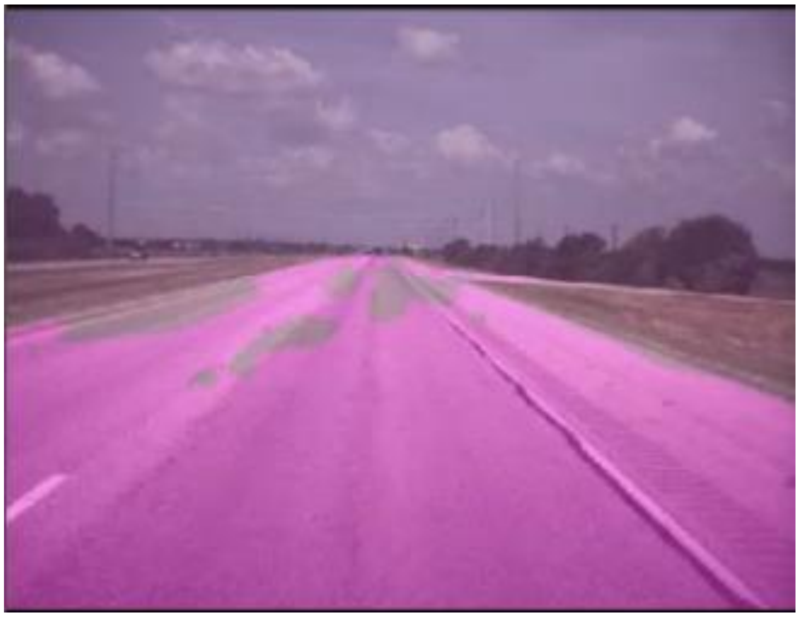


Figure 10: combined model

In Figure 10, the combined models output shows several patches of the road as background which is not correct.

The combined LaneAtt model also tends to show false positives, to the sides of the actual road for gravel roads, while for marked roads showing the emergency lane as its own viable lane. Examples are shown in Figures 11 and 12.

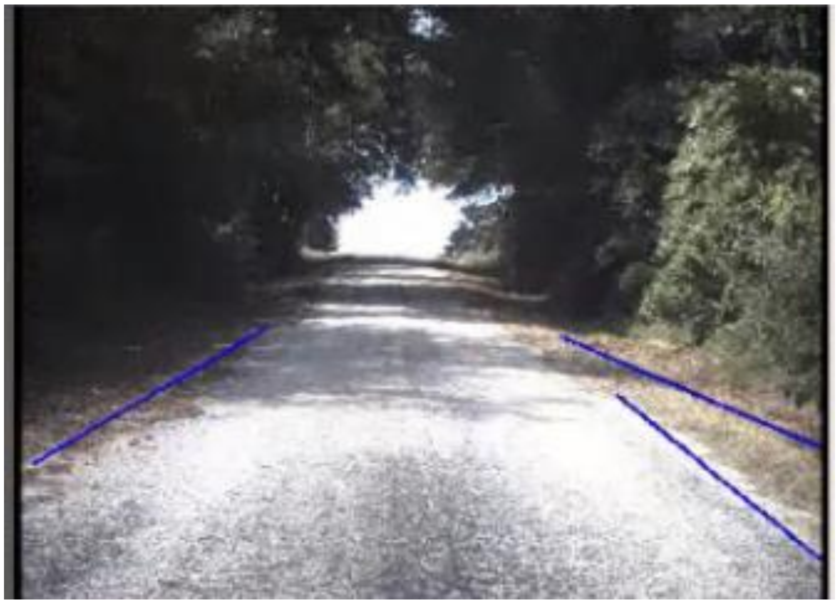


Figure 11: detected lane next to gravel road, combined model

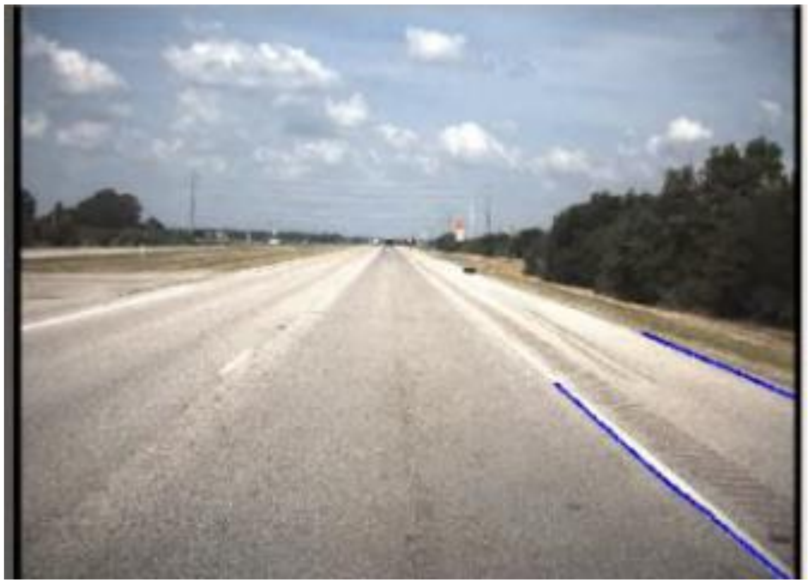


Figure 12: Emergency lane, combined model

Both approaches struggle with low light situations (under-exposure), for example if shade is present on the road or the sun is not shining anymore. This is represented in Figure 13, where no Lanes were detected in either approach.

Similarly, they also struggle with situations where the camera is overexposed, as the edge of the road is not discernible from surrounding areas. This Could be an artifact of the annotated data being recorded in similar light conditions, and a more variable dataset would benefit both approaches, likely solving this issue. A different camera also may be less prone to over/underexposure and post-processing of the images could enhance the results likewise. Interestingly, segmentation seems to be much less prone to having detection issues with overexposure than the lane detection.

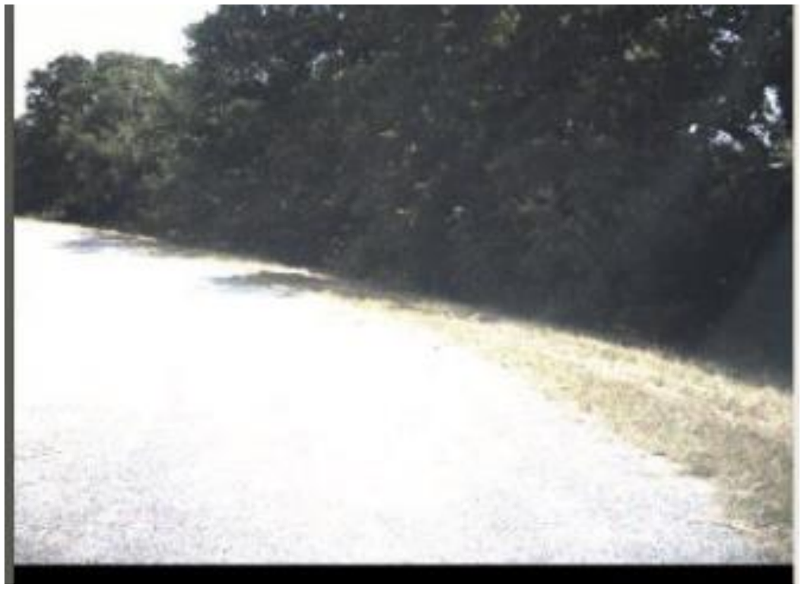
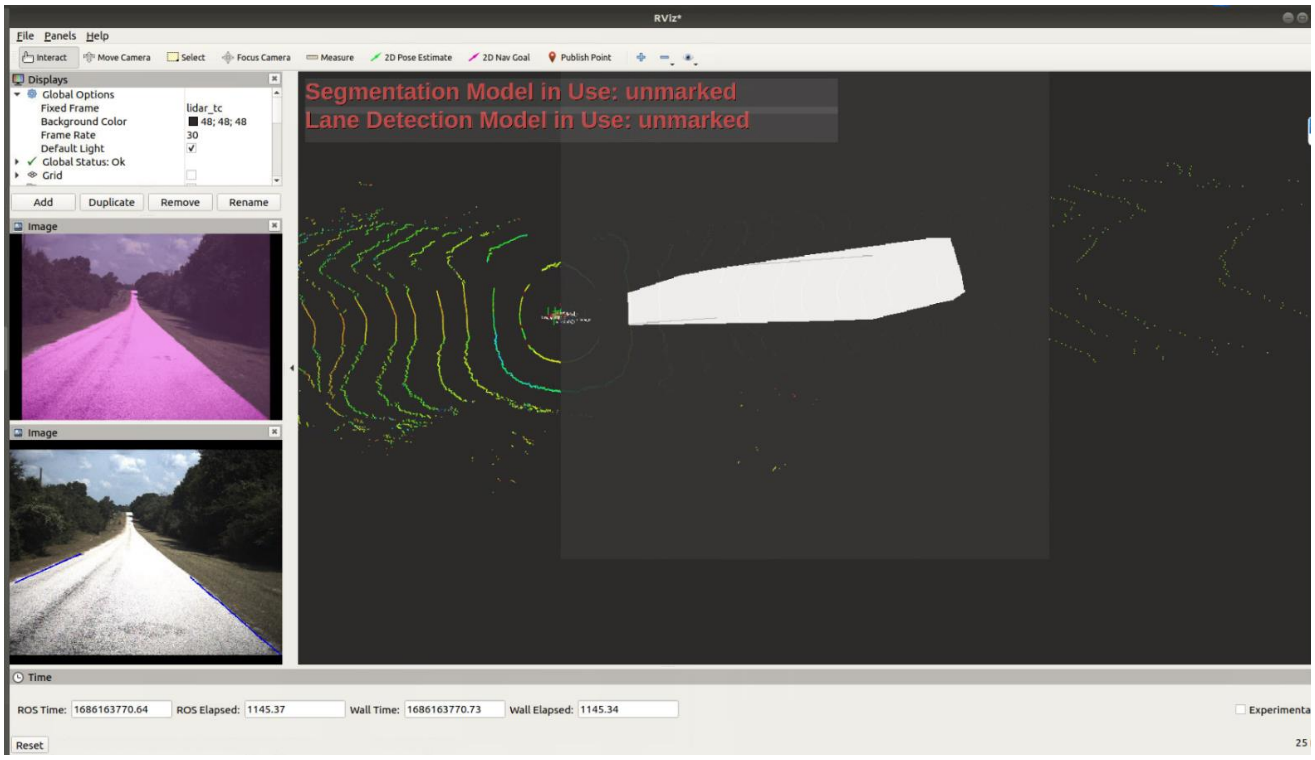


Figure 13: Overexposure, no detection, same for combined/separated models

One issue of the Lane Detection model is that it is sometimes uncertain about which environment it is in at the time, resulting in no or low quality detections for prolonged periods of time as the models are constantly switching. During these times, the combined model does perform better but shows false positives.

The generally accepted understanding is that a larger dataset can lead to better generalization, meaning a model will be able to perform better on unseen data as it is able to learn more patterns. Accuracy is likely to improve as well with a larger dataset as it can learn more details about different examples. Robustness will generally also improve along with lower (better) variability between the predictions.

One caveat is that larger datasets need to retain high quality annotations and images. If a dataset is heavily biased or has noisy data - the resulting model may not perform better than a smaller dataset with higher quality data. Overall, we believe our approach would benefit from a larger dataset (~5000 Annotations) and still prove viable. The dataset should include images from different days, light conditions and various other roads to provide greater training variety. The hope of being able to train these models with a large dataset is that the concept of generalization applies (as mentioned above) meaning that the model can infer where the road or lane markings are located even though it has never seen anything like this road in the training data. Although that is the hope in theory, in practice we have not seen this work. Even when using large datasets such as CULane for Lane Detection, 133,235 images & annotations, and Cityscapes for Segmentation, ~25,000 images & annotations, or other large datasets. Generally, the models will work fairly well on the specific environment they have been trained on but on our roads (Texas Highways or rural and gravel roads) these models lack accuracy. With the proposed switching algorithm, even small datasets can produce minimally reasonable accuracy. Looking at this work, with comparatively small datasets of image annotations for each lane detection and road segmentation, divided roughly 50/50 between marked roads and unmarked roads, the models are effective at detecting the type of roads/lane markings that they were trained on but has little generalization, as unseen roads generally show poor detection results.



Segmentation Model in Use: unmarked
Lane Detection Model in Use: unmarked

```
Global Options
  Fixed Frame        lidar_tc
  Background Color   48; 48; 48
  Frame Rate         30
  Default Light      ✓
Global Status: Ok
Grid
```

```
ROS Time: 1686163770.64   ROS Elapsed: 1145.37   Wall Time: 1686163770.73   Wall Elapsed: 1145.34
```

Figure 14: Full system visualization

Finally, Figure 14 is the entire system view together as it runs with full visualization on the vehicle. On the left side one can see the image output of the segmented road (top image) and detected lanes (bottom image). The right side shows a top down view of the vehicle, the colorized ground-segmented pointcloud overlaid with the costmap that shows the two detected lane markings in gray along with the segmented road in white while everything around is marked in black. Additionally, a text overlay in the upper left corner shows the current model in use for each image detection algorithm.

CONCLUSION

Summarizing the above presented system, given a small dataset we show an overall better functioning system than a traditional approach of training one model with all the data. Based on our analysis, the split model system performs at a higher accuracy in most cases with fewer false positive detections than the combined model. The cost of running two models at a time is a slightly higher resource utilization on both the CPU and GPU but with today's hardware this is generally not a constraint, as many other detection algorithms will require more resources than our entire system combined. This system produces reasonably accurate results with minimal annotation effort, and small resource requirements. In those instances, this system will outperform standard approaches in most situations. Using a similar approach with extremely large datasets could still prove that the system works but at that point a model may be close enough to generalization that this system would be redundant. One of the major benefits we see is that knowing what type of road you are on may also give hints about how the vehicle should behave, such as appropriate speeds that may be lower than the actual speed limit, expected grip levels and/or where within the road you should be traveling.

FUTURE WORK

Besides just annotating more data, or improving data/annotation quality, data augmentation should be implemented. Data augmentation is a process in which the original data is slightly modified (flipped, rotated, noise added, changed contrasts, brightness etc.) in order to provide a larger training variety.

Additionally, this system would likely benefit from using simpler model or backbone architectures for the segmentation as well as the lane detection. While generally speaking a simpler model/backbone may infer faster, it usually comes at the cost of losing some accuracy. In contrast, given a small dataset, it may be beneficial to use a simpler model/backbone to prevent overfitting - overfitting is the opposite of generalization, meaning the model "learns" noise in the training data more than features that may actually define the object of interest. Models that have more parameters, being higher complexity models or backbones, are prone to overfitting with small datasets.

LaneAtt should be replaced as it has trouble detecting lanes accurately in low radius curves due to the way it interpolates points (based on a starting point and an angle it will create equidistant points that are generally of a more linear nature), this results in certain lanes to be showing across the image even though this is not true. What we have seen is that these artifacts are partially removed through fusion with the 3D Lidar points; regardless, it should still be addressed.

Finally, rewriting the code to run faster/use less resources in general. Lane Detection can run extremely fast, while Road Segmentation is inherently slower as it looks at every pixel and classifies each pixel whereas lane detection (or image detection in general) only classifies small areas/patches of the image. There are several potential ways of optimizing runtime/resource usage. The system is heavily dependent on pytorch, other similar libraries such as tensorflow,tensorrt or ONNX (Open Neural Network Exchange) may reduce cpu utilization while also reducing inference time. This would come at a slight trade off as for certain libraries the model may need to be converted, which can result in modifications to the model and performance may vary.

Fusing the detection results from Lane Detection and Road Segmentation is rather simple, but especially the Road Segmentation takes a lot of time to compute due to a large amount of pixels having to be matched with the lidar point cloud. The current implementation uses Alpha Shape to compute the convex Hull of the road area (The outermost boundary of the detected area, and any points within) which in itself requires too much time and cannot be sped up easily. Using concave hulls, the resulting road area is more accurate but cannot be computed in a reasonable amount of time.

REFERENCES

[1] *Tesla Autopilot Using HD Maps? - AutoPilot Review*. AutoPilot Review. (2022). Retrieved 12 September 2022, from https://www.autopilotreview.com/tesla-autopilot-hd-maps/.

[2] Tabelini, L. (2020). *lucastabelini - Overview*. GitHub. Retrieved 12 September 2022, from https://github.com/lucastabelini.

[3] Tabelini, L., Berriel, R., Paixao, T., Badue, C., De Souza, A., & Olivera-Santos, T. (2021). *Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection*. Openaccess.thecvf.com. Retrieved 8 September 2022, from https://openaccess.thecvf.com/content/CVPR2021/papers/Tabelini_Keep_Your_Eyes_on_the_Lane_Real-Time_Attention-Guided_Lane_Detection_CVPR_2021_paper.pdf.

[4] Zermas, D., Izzat, I., & Papanikolopoulos, N. (2017). *Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications*. Ieeexplore.ieee.org. Retrieved 12 September 2022, from https://ieeexplore.ieee.org/document/7989591.

[5] Yan, Y., Duckett, T., & Bellotto, N. (2017). *Online learning for human classification in 3D LiDAR-based tracking*. Ieeexplore.ieee.org. Retrieved 12 September 2022, from https://ieeexplore.ieee.org/document/8202247.

[6] *GitHub - open-mmlab/mmsegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark.* GitHub. (2020). Retrieved 12 September 2022, from https://github.com/open-mmlab/mmsegmentation.

[7] Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). *Pyramid Scene Parsing Network*. Arxiv.org. Retrieved 12 September 2022, from https://arxiv.org/pdf/1612.01105.pdf.

[8] Ort, T., Paull, L., & Rus, D. (2018). *Autonomous Vehicle Navigation in Rural Environments without Detailed Prior Maps*. Toyota.csail.mit.edu. Retrieved 12 September 2022, from https://toyota.csail.mit.edu/sites/default/files/documents/papers/ICRA2018_AutonomousVehicleNavigationRuralEnvironment.pdf.

[9] Ort, T., Jatavallabhula, K., Banerjee, R., Gottipati, S., Bhatt, D., & Gilitschenski, I. et al. (2019). *MapLite: Autonomous Intersection Navigation Without a Detailed Prior Map*. researchgate.net. Retrieved 12 September 2022, from https://www.researchgate.net/publication/338060625_MapLite_Autonomous_Intersection_Navigation_Without_a_Detailed_Prior_Map.

[10] Guezel, M. (2013). *Autonomous Vehicle Navigation Using Vision and Mapless Strategies: A Survey*. journsals.sagepub.com. Retrieved 12 September 2022, from https://journals.sagepub.com/doi/full/10.1155/2013/234747.

[11] Jeong, J., Yoon, T., & Park, J. (2018). *Towards a Meaningful 3D Map Using a 3D Lidar and a Camera*. ncbi.nlm.nih.gov. Retrieved 12 September 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6111277/.

[12] Wu, Z., Qiu, K., Yuan, T., & Chen, H. (2021). *A method to keep autonomous vehicles steadily drive based on lane detection*. journals.sagepub.com. Retrieved 12 September 2022, from https://journals.sagepub.com/doi/full/10.1177/17298814211002974.

[13] *GitHub - AutoLidarPerception/segmenters_lib: The LiDAR segmenters library, for segmentation-based detection*. GitHub. (2020). Retrieved 12 September 2022, from https://github.com/AutoLidarPerception/segmenters_lib.

[14] *Unsupervised Llamas Lane Marker Dataset*. Unsupervised-llamas.com. (2019). Retrieved 12 September 2022, from https://unsupervised-llamas.com/llamas/index.

[15] Pan, X., Shi, J., Luo, P., Wang, X., & Tang, X. (2018). *Spatial As Deep: Spatial CNN for Traffic Scene Understanding - CULane Dataset*. Xingangpan.github.io. Retrieved 12 September 2022, from https://xingangpan.github.io/projects/CULane.html.

[16] *TuSimple/tusimple-benchmark*. GitHub. (2017). Retrieved 12 September 2022, from https://github.com/TuSimple/tusimple-benchmark/wiki.

[17] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). *Vision meets Robotics: The KITTI Dataset*. Cvlibs.net. Retrieved 12 September 2022, from https://www.cvlibs.net/datasets/kitti/.

[18] *A Case Study: Edge Detection Techniques Using Hough Transform and Canny Edge Algorithm*. (n.d.). Docslib. Retrieved September 12, 2022, from https://docslib.org/doc/6340679/a-case-study-edge-detection-techniques-using-hough-transform-and-canny-edge-algorithm

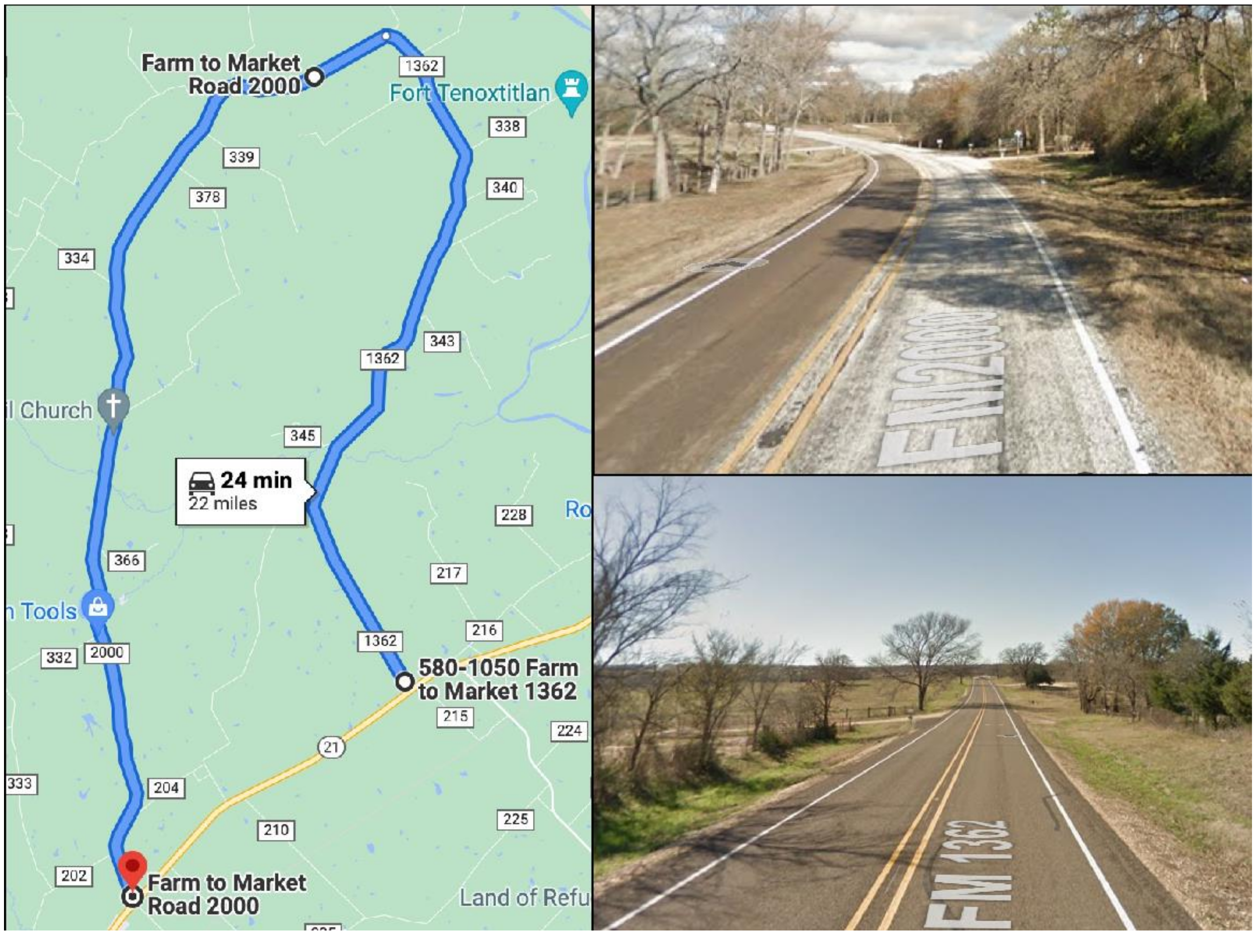
[19] Redmon, J. (2018).*YOLOv3: An Incremental Improvement*. arXiv.org. Retrieved September 12, 2022, from https://arxiv.org/abs/1804.02767

[20] Noury, C., Teulière, C., & Dhome, M. (2020). *Light-field camera calibration from raw images*. hal.uca.fr. Retrieved September 12, 2022, from https://hal.uca.fr/hal-01657735/document

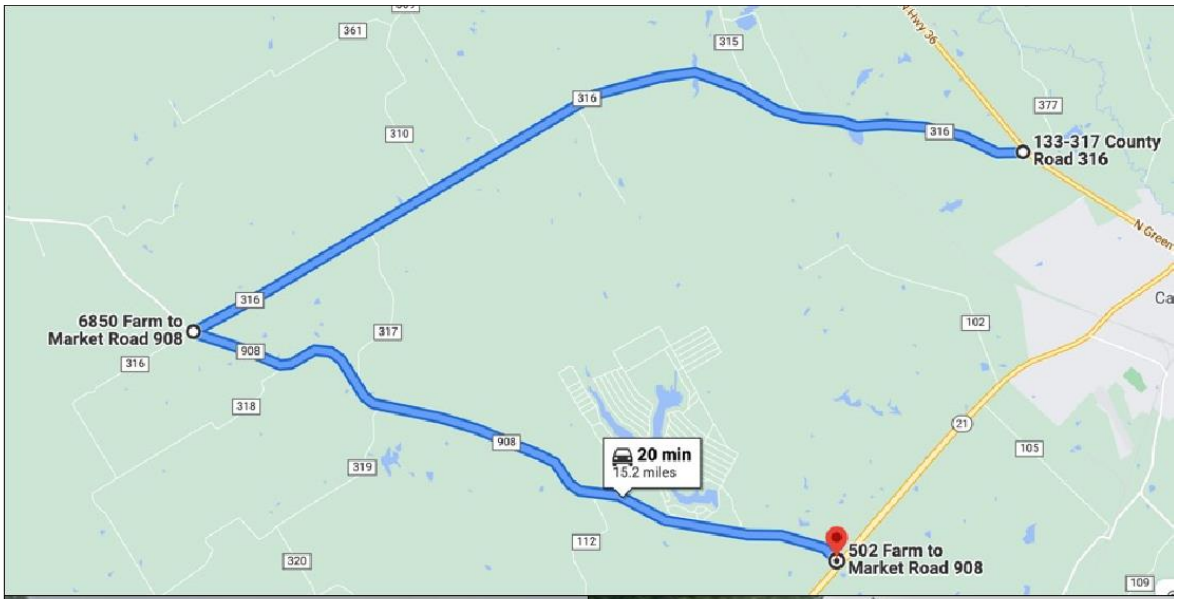
[21] Marti, E., Perez, J., de Miguel, M., & Garcia, F. (2019). *A Review of Sensor Technologies for Perception in Automated Driving*. dsp.tecnalia.com. Retrieved September 12, 2022, from http://dsp.tecnalia.com/bitstream/handle/11556/778/08846569.pdf

[22] Sekachev, M., Manovic, N. (2023). *Cvat.* Retrieved 12 June 2023, from https://www.cvat.ai/,

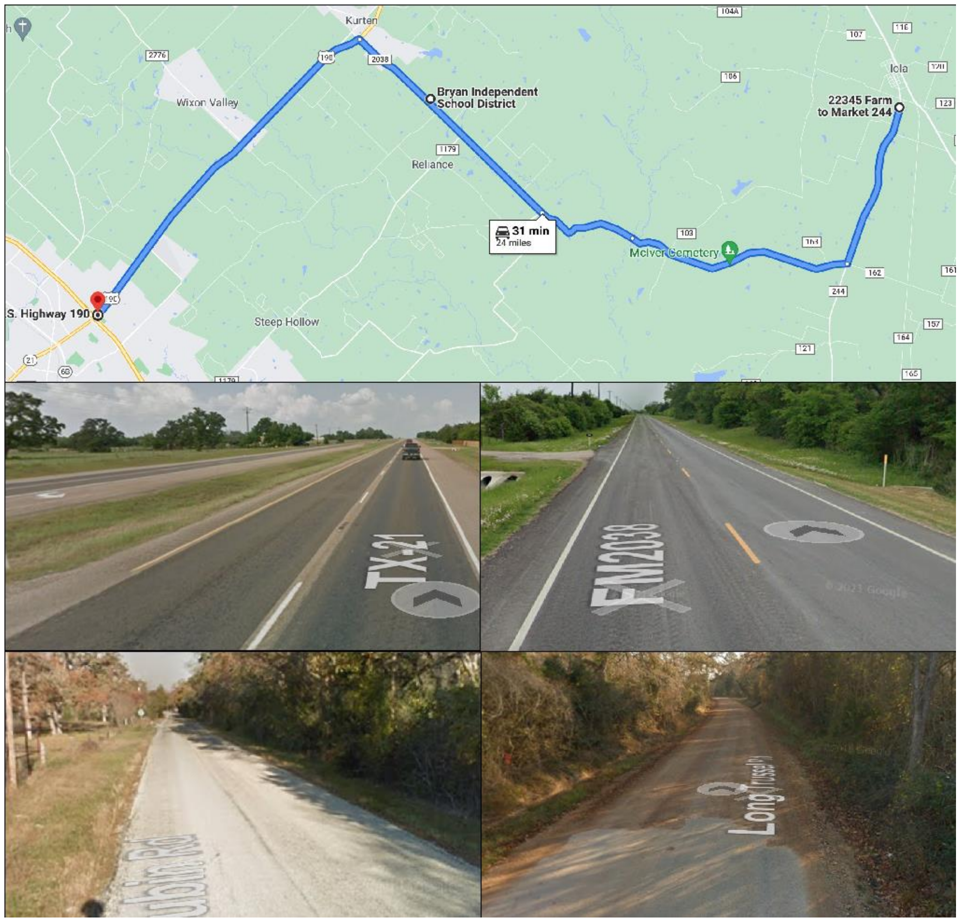
APPENDIX



Test Route #1, Farm to Market Roads 1362 and 2000. This 22-miles stretch of two-lane road offers a range of medium to high-quality pavement and marking. There are over 25 connections to local roads through at-grade intersections.



Test Route #2, Farm to Market Road 908 and County Road 316. This 15.2-miles stretch of road offers high-quality pavement and marking (on Farm to Market Road 908) and low-quality pavement with no marking (on County Road 316). There are 31 connections to local roads through at grade intersections, including a two-way stop sign controlled intersection.



Test Route #3, US Highway 190 and Farm to Market Roads 2038 and 244. This 24-miles stretch of road offers high-quality pavement and marking on a divided highway (on US Highway 190), both high-quality pavement and marking (on Farm to Market Road 2038), low-quality pavement with no marking (on Farm to Market Road 2038), and unpaved gravel road (on Farm to Market Road 244). There are over 60 connections to local roads through at grade intersections, including multiple stop sign controlled intersections.