Infrastructure-based Detection and Localization of Road Users for Cooperative Autonomous Driving

Author: Lance Bassett
Advisor: Henry Liu

Introduction

Ever since DARPA's Grand Challenge in 2004, autonomous vehicles (AVs) and their adjacent technology have been the beneficiaries of billions of dollars as companies try to jam as much computing and sensing capability onto the vehicle as possible[1]. Despite those resources, AV technology has remained quite costly, and the timeline for anyone releasing a truly autonomous vehicle–one that you can feel comfortable sleeping in–has been constantly extended. The road is simply too complex–and the technology required too expensive–for the vehicle to be the sole observer and processor of its environment[2]. Both of these problems can be addressed by off-loading computation and perception to the infrastructure.

Our roadside units–equipped with cameras, LiDAR, and computational resources–can make thin-client AVs (AVs with fewer sensors and less computational power) viable, reducing the cost per car and improving safety at the same time. Residential, highway, and open-road scenarios are easily manageable by many AVs being tested today. It's the dense city centers, examples of which are seen in Figure 1, where parked cars line the roads and pedestrians cross the street at will that require multiple layers of sensor redundancy and lightning-quick computation, driving on-vehicle costs way up. Fortunately, these difficult environments also tend to have a significant amount of infrastructure in place. Our roadside units can be placed in difficult environments/intersections to take on some computational burden and can perceive and warn nearby vehicles of potential hazardous behavior, or even stop cars to avoid it completely. It also appears that there will inevitably be a long transition period where vehicles are "connected" (communicating with infrastructure and other vehicles) while not being fully autonomous[4]. Our system could be used by both connected and autonomous vehicles to improve safety during this transition.

Figure 1: Examples of difficult or blind intersections in which our units could be placed. These are intersections where it is difficult or dangerous to only perceive things from the field of view of the vehicle.

In order to capitalize on the safety and cost potential of this technology, the units must be able to reliably perceive and understand what is happening in their environment, which is where our research (and my capstone project) comes in.

This project focuses on the methods for detection–finding an object's 2D position in an image–and localization–finding an object's 3D position in the world–of vehicles and pedestrians as they move through the scene.

Each roadside unit is positioned about 20 meters off the ground, angled toward the ground, and ideally positioned to be able to view a blind corner or some other less visible part of the intersection/sidewalk. Our methods will use the unit's onboard camera and processor in the future, but we are currently using simulated data in order to have access to a sufficiently large amount of data and to streamline the early stages of research. Figure 2 shows an example view of one of the simulated units.

Figure 2: View of our simulated intersection

Formally, this project's goal is a lane-level precise detection and localization system. In other words, we want to be able to palace things in the scene with enough precision that we can tell what lane they're in. Average localization error between 0.3 and 0.5 meters would be acceptable, and anything lower we would consider exceptional. By solidifying these methods, we will have solved a significant piece of the puzzle in building a comprehensive intersection supervision system. In the grand scheme of things, this is a step towards greatly improving pedestrian and driver safety in some of the most dangerous spaces on the road.

This project uses deep learning to accomplish our detection task, adapting existing architecture from a model called CenterNet[4]. For localization, we use a precomputed coordinate mapping from pixel location to world location based on our knowledge of the scene. Both of these methods capitalize on the fact that we have the uncommon advantage of a fixed scene and static background due to the nature of infrastructure-mounted sensors. It allows us to make certain assumptions about where things are located in the scene, providing unique opportunities for detection and localization methods that will be described later.

The rest of this report will go into detail on the dataset, deep learning models, and additional methods we use to get our results. I will then present and discuss the results, compare alternate approaches

we've used, describe the advantages and disadvantages of our approach, and conclude with future work we plan to pursue.

Summary of Approach

Our approach uses two neural networks, trained separately, and then run back to back at prediction time. The first predicts the center of the objects and classifies them (as vehicle or pedestrian), then, for each vehicle we've predicted in the image, we create a small crop of the image around the vehicle and pass it into our second network. The second network predicts the bottom corners of each vehicle, analogous to how the first network predicts the centers. This allows us to obtain a footprint on the road for each vehicle. To localize, we simply look at our coordinate map at the same pixel locations we've detected vehicle corners and pedestrians to obtain our global *x* and *y* coordinates. The step-by-step pipeline is visualized below in Figure 3, and details about our architecture and processes are found later in the report.

Figure 3: Step by step visualization of the pipeline.

Detailed Methods

This section will go into detail about each step in our development process, including our dataset, architecture, training process, and other methods we use to improve results.

Dataset Creation

As mentioned above, we have been developing our method using simulated intersection data. We use the CARLA driving simulator to generate images and labels of an intersection. For the results in this report, we save only every 20th frame of simulation to avoid similar images and use UQI [6] to compare consecutive frames to further prune our dataset.

To train our second model, we also generate a dataset of crops by cropping the images from our full-size dataset. It's important that crops contain a vehicle completely in order for the model to predict its corners, but we also don't want to confuse the model by including other vehicles or an unnecessary amount of background. For this reason, each crop must change according to the size of the car in the image. We take the simple approach of basing crop size on the vehicle's y position in the image–the further down a vehicle appears in the image, the closer it is to the camera and the larger it appears. We've arrived at the following functions for crop width (c_w) and crop height (c_h) through empirical means. Samples of the vehicle crops using these equations are displayed in Figure 4.

$$c_w = 0.02y^{3/2} + 10 \qquad c_h = 0.015y^{3/2}$$

Figure 4: Samples of vehicle crops. The center of the crop is the vehicle's bottom center.

Network Description and Architecture

The architecture for both of our models is heavily based on CenterNet. CenterNet is a fully convolutional method that represents objects by their center points, as visualized in Figure 5, and generates "heat maps" to predict object location, size, depth, position, or, in our case, class.

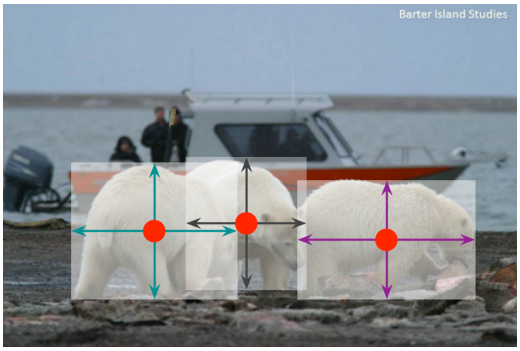

Figure 5: CenterNet represents objects by their center, then estimates other necessary information like depth and size.

The primary output of our model is a probability map: an "image" the same size as our input where each pixel's value is the probability that an object center appears at that pixel. We extract local maximums from the map to find the locations where centers are most likely. The general idea for detection is shown below in Figure 6.

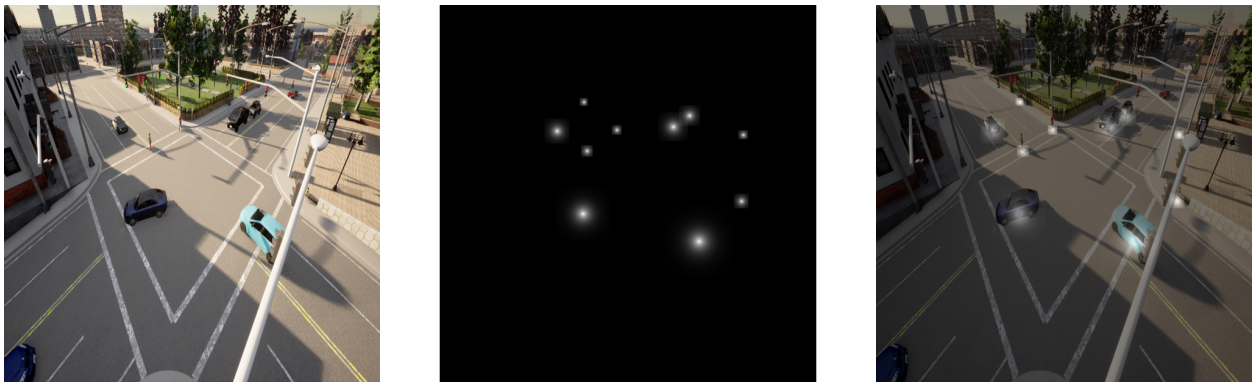

Figure 6: Visualization of center probabilities. Left is our RGB image, middle is the corresponding ground truth probability map, and right is the two overlapped.

Network 1. The input to this stage is a full-size image (HxWx3). We use an encoder-decoder and feature pyramidal network (FPN) with a ResNet18 backbone to generate a feature map, then pass that to our center and classification heads. After extracting the local maximums from our center heatmap (HxWx1), we index the classification heatmap (HxWx3) at each predicted center to obtain the predicted class of the objects. This process is visualized in Figure 7.

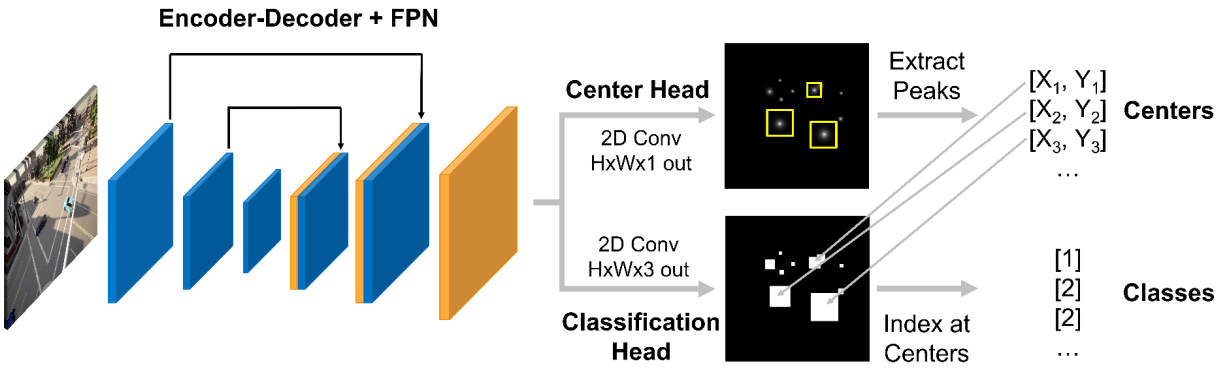

Figure 7: Architecture of our first network. Extracted centers are used to index additional maps.

Network 2. The input to this stage is a cropped image based on the vehicle centers of the previous stage. We use the same architecture as the first network to generate a feature map, then pass that to a corner prediction head. Assuming the cropped image size is MxN, this network outputs an MxNx4 heatmap where each channel acts similarly to the center probability heatmap from the first stage, but each channel is associated with a specific corner of the target vehicle. In other words, the pixel values in the first channel of the output are the probability that the front-left corner of the target vehicle appears at that pixel. The second, third, and fourth channels predict analogous information for the back-left, back-right, and front-right corners. By standardizing this in our labels, we can obtain an ordered set of corners with semantic meaning as opposed to an unordered group of four points, which provides the ability to complete missing corners down the road. This whole process is visualized in Figure 8.

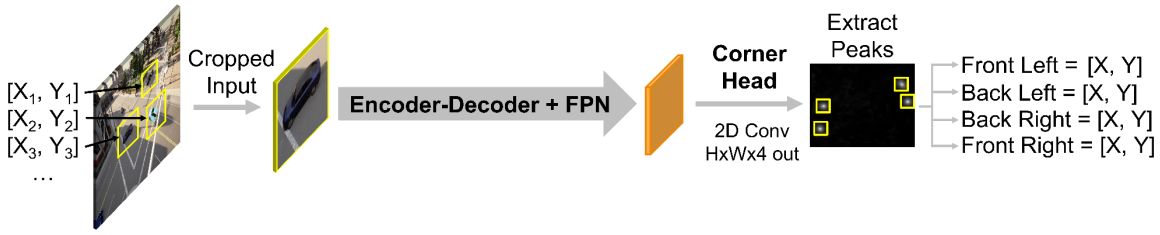

Figure 8: Architecture of our second network. We use the same encoder-decoder architecture as before with a new output head for corners.

Coordinate Map Creation

Detection of objects is important, but without quality localization, even a perfect detection system doesn't mean much. Localization is not done by the network in our case. We instead use a pre-generated

coordinate mapping (equal in size to our input image, two channels) to translate between pixel location and world location. More concretely, once you have generated the coordinate map and have the pixel location of a vehicle center in the image, the value of channels one and two of the coordinate map at that pixel will be the world x and y coordinates, respectively.

We generate these coordinate mappings by fitting a homography with RANSAC[6] to a set of correspondences of pixel and world locations, i.e. ground truth matches between a place in the image and a place in the world. This can be thought of as finding a "plane of best fit" to the road surface. In our case, corresponding pixel and world points are automatically obtained via the simulator. In a live deployment, we plan to precisely map pixel locations of notable road features to their GPS location.

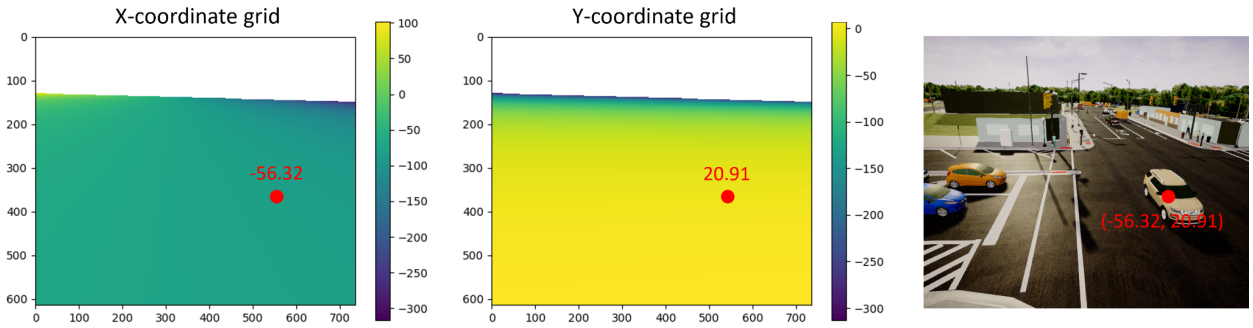

Figure 9: Visualization of the location maps that are used to obtain corresponding global 3D coordinates from a pixel location.

Corner Completion

While not common, it is sometimes the case that our second network is only able to predict 3 corners of a vehicle. These failures usually occur for the back corner of the vehicle (hidden by the vehicle itself) or for a corner that is obstructed by another object. Because the output of our second network is ordered, however, we know which predicted corners are which (front left, back right, etc.) and, consequently, we know which corner we are missing when we are only able to extract 3. Using the center and opposite corner, we can complete the bounding box and salvage the detection, as seen in Figure 10.

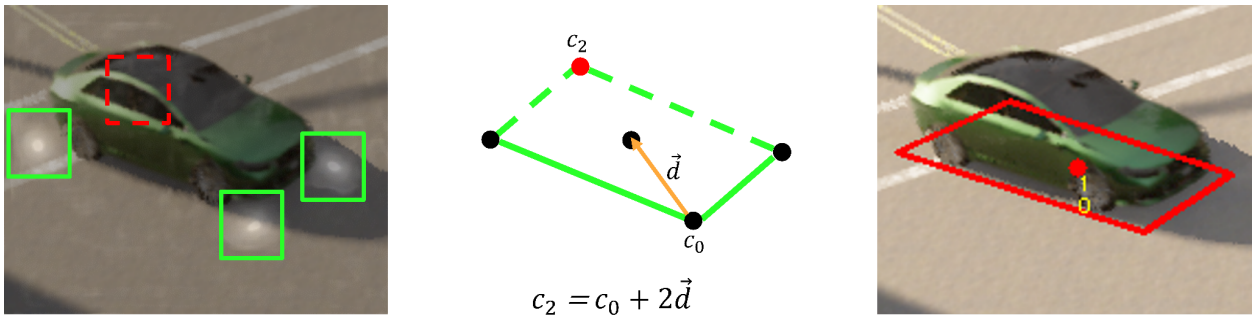

Figure 10: Example of vehicle missing corner and subsequent completed result

Training

We train our first network on 4.2k examples, and our second on 35k cropped examples. We optimize both center heatmaps and corner heatmaps with TopK MSE loss and use Cross-Entropy Loss for our classification maps.

Center Jitter Augmentation. We also employ a "jitter" data augmentation to our second network dataset. This involves adding Gaussian noise $N(0,\ \sigma^2)$ to the centers on which we base our vehicle crops in order to produce an "off-center" crop. The goal of this augmentation is to make it so that poor center predictions from the first network don't ruin the results of the second stage, which depends on the crops from the first. By including off-center crops in our dataset, we teach the second model how to deal with them when received by the first network. The results for this report are trained on a dataset with $\sigma^2 = 2$. A visualization of the augmentation is provided in Figure 11.

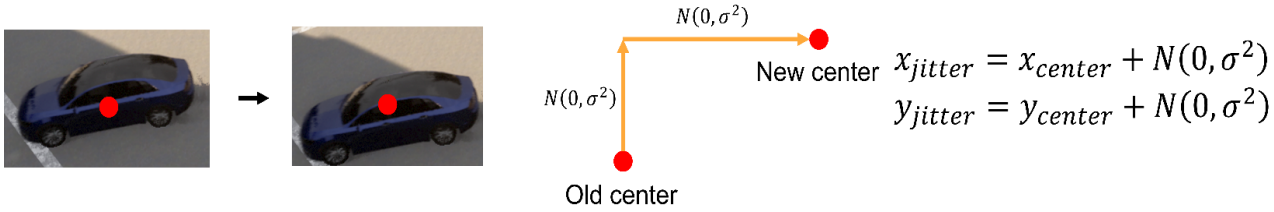

Figure 11: Example of jitter augmentation. Center shift changes the resulting crop.

Results

Tables 1 and 2 below summarize our results alongside some visualizations in Figure 12 of our resulting detections. The results shown are averages across each image in our validation dataset, which consists of 1936 frames of intersection data. Our goal for localization error was anywhere between 0.3 - 0.5m, so achieving 0.213m in our results was exceptional. This quality result comes from improvements in the coordinate maps as well as in the accuracy of our detections–a more accurate pixel location leads to a more accurate result from the coordinate map.

We compare our best model to a "single stage" model. This was an earlier iteration of our model that predicted centers, corners, and classes at once based on the entire image. It worked well enough, but our initial motivation in opting for a two stage model was its simplicity, which the single stage model lacks (discussed in greater detail in the "Advantages" section).

Method	Center Pixel Error	Corner Pixel Error	Corner Global Error
Two Stage	**2.939px**	**2.02px**	**0.213m**
Single Stage	3.743px	3.851px	0.259m

Table 1: Errors for our two different methods of detection. Single stage predicts corners and centers in the same step.

Method	Missed (n = 1936)	Detection Rate	Fixed
Two Stage	**21**	**0.989**	**9**
Single Stage	48	0.957	25

Table 2: Detection rate and completion statistics for our two different methods of detection.

Figure 12: Sample results from the two stage model. First stage trained on 4.2k examples, second stage trained on 35k examples.

As mentioned earlier, our best model was trained with the jitter augmentation ($\sigma^2 = 2$). The benefits of jitter are difficult to see on our simulated dataset because there is so little noise and our centers are predicted very well without jitter. We can demonstrate the additional robustness that jitter provides by adding increasing amounts of noise to our predicted centers at runtime and comparing the second network results on the resulting crops. The results at 4 different levels of noise ($\sigma^2 = 0, 2, 4, 6$) are shown in Figure 13. The disparity in all metrics between the jitter model and baseline model grows significantly as noise increases, and jitter continues to perform well even on noise levels higher than what the training dataset was augmented with. Real-world data is much messier and noisier than

simulated data and we expect to take a slight performance hit when transferring to a real scenario. For this reason, it is important that the second stage is able to perform well despite errors in the first stage.

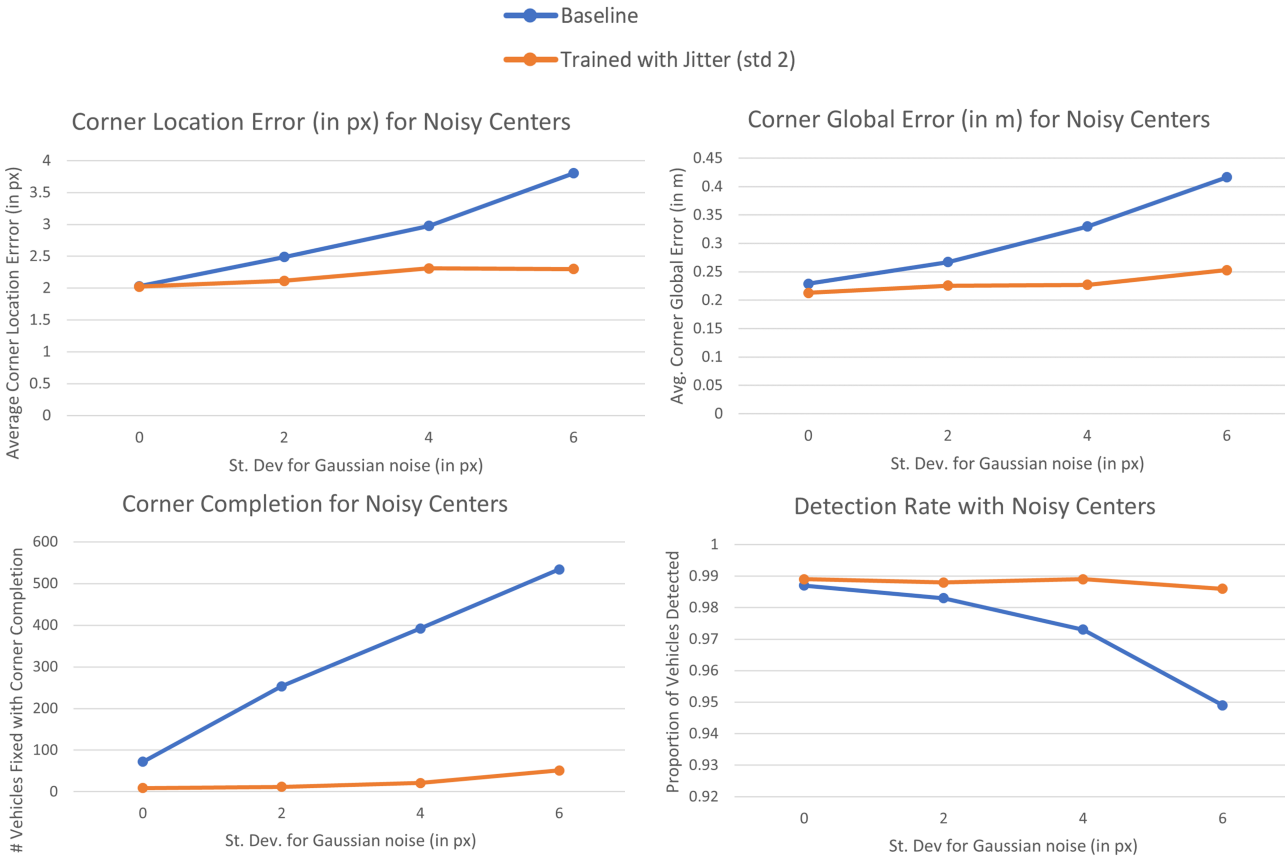

Figure 13: Graphs comparing model performance between jitter and no jitter. Simple augmentation makes the second stage robust to imperfect centers in the first stage.

Discussion

The results we've obtained have been exceptional and we believe our methods show a lot of promise as we look toward applying them to real-world data. Highly performant detection systems already exist and have been deployed, but a major issue that we've tackled with our method is the accurate, repeatable localization of our detections in a way that takes advantage of the fixed scene. We solve this with a combination of accurate detections and quality coordinate maps. The answers we've found through our research are all part of solving the larger puzzle of a comprehensive and cooperative intersection supervision system.

Advantages

Our current approach has multiple advantages, some of which are born out of our two stage cropping approach (as opposed to one stage).

The first advantage of our approach is its simplicity, both in general and relative to a single stage model. The labels and results are visual, and the approach is split up into two logical steps. A major complication in our single stage approach was the fact that corners predicted by the model are not inherently associated with a single vehicle in the image. This led to a complex corner-to-vehicle matching process that required an additional prediction head and was prone to mistakes. By cropping around each vehicle, we are implicitly associating the predicted corners with the vehicle in the crop, removing the need for a matching process.

Another benefit to training our second network on crops is that it is more universally deployable than a model trained on the entire view of the intersection. In other words, we could train a model on crops from one or a few intersections and deploy it to many others, requiring less data and training time. This benefit comes from the fact that crops exclude the variability found in the background and surroundings of the scene. Figure 14 demonstrates this point–the intersections are quite different, but the crops are quite similar.

Figure 14: Comparison demonstrating low variability in crop of scene, despite high variability of actual scene.

Having two stages can also allow the second stage to correct the first. We can predict redundant information in the second stage and discard detections for which there is a disagreement to avoid false positives. For example, if the first stage network thinks that a fire hydrant is a vehicle, we will extract a center and pass a crop of the fire hydrant into the second stage, which can then classify the crop *again*, giving the model a second chance at correctly classifying the fire hydrant as "background." I will note that this has not been tested and is still just a theory, but it has significant potential and is still worth mentioning.

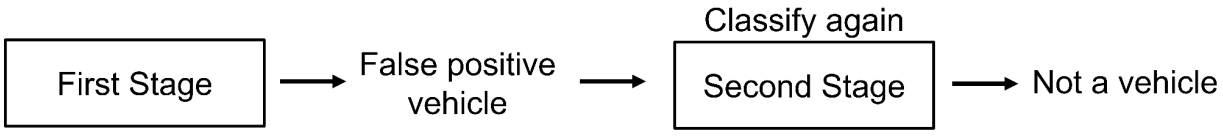

Figure 15: Diagram illustrating two stage refinement. False positives can be detected and thrown out.

The last advantage I will mention is the pre-processed nature of our coordinate maps. Because the map is generated offline, it is not a problem to sacrifice time for accuracy. We can use an extremely complex method to map out the surface of the road (within reason) without incurring any speed penalties at runtime–all we have to do is read from our existing coordinate map. In the early stages of development, we attempted to have the model predict the global x, y, and z coordinates of objects and our best results yielded an error of 0.327m. Reasonably accurate, but this is a very difficult, unconstrained task for the

network to perform and we would expect a significant performance hit when working on real data. Additionally, there is much less potential for improvement compared to the simple, yet effective coordinate map method.

Disadvantages

There are, of course, things to note about our approach that are not perfect. We do believe that the positives greatly outweigh the negatives, but the latter is still worth considering.

Our model requires two separate datasets (one uncropped, the other cropped) for training, and the cropped dataset comes with additional complications. You can space out the time interval between frames to avoid duplicates easily enough, but individual cars may sit still for a very long time resulting in duplicate crops despite a large time interval. To avoid throwing away many frames of useful data, we use an image hashing process to remove duplicates from our cropped dataset, which adds complexity and time to the dataset creation process and is admittedly still not perfect.

Additionally, although we have proposed the jitter augmentation to lessen its impact, there is still the fact that the two networks in our setup are heavily dependent on each other. Mistakes in the first stage will propagate to some degree and both models need to be highly performant to obtain good results, which can be difficult with limited data.

Conclusions

The methods detailed in this report show significant potential for a lane-level detection and localization system, but I must emphasize that we have been working exclusively with simulated data. Until we have comparable results on real-world data, we have very few conclusions about the real-world utility of our methods. Additionally, until we compare more comprehensively with other architectures and methods, we can only make conclusions about the effectiveness of our approach in a vacuum, relative to itself and our past approaches. While it's difficult to conclude anything concrete (*this* is better than *that*), we can say with confidence that we have devised a method with significant potential to provide detection and localization results to an intersection supervision system. To conclude anything further, we will rely on the results of our future work.

Future Work

I've mentioned many additional steps we need to take throughout the report, and the most important of those is the transition to real-world data. Barring any catastrophes, results are good enough right now that we would expect to be within our goals even on real-world data. We are also looking at more robust methods of determining vehicle crop size. We want to explore a crop size "map", similar in theory to our coordinate map, which we can use like a lookup table for each pixel location in the image to obtain a crop size. Lastly, as mentioned above, would be a comprehensive comparison between our method and existing methods with the ultimate goal of publishing a paper on our methods. Future work for this project would expand beyond detection and localization toward fusion between multiple sensor systems and live deployments in real intersections.

References

1. “The DARPA Grand Challenge: Ten Years Later,” *DARPA RSS*. [Online]. Available: https://www.darpa.mil/news-events/2014-03-13. [Accessed: 19-Oct-2021].

2. “What is vehicle-to-infrastructure (V2I) communication and why do we need it?,” *3M in the United States*. [Online]. Available: https://www.3m.com/3M/en_US/road-safety-us/resources/road-transportation-safety-center-blog/full-story/~/what-is-vehicle-to-infrastructure-v2i-communication-and-why-do-we-need-it/?storyid=021748d7-f48c-4cd8-8948-b7707f231795. [Accessed: 19-Oct-2021

3. D. Gessner, “Experts say we're decades from fully autonomous cars. Here's why.,” *Business Insider*, 22-Jul-2020. [Online]. Available: https://www.businessinsider.com/self-driving-cars-fully-autonomous-vehicles-future-prediction-timeline-2019-8. [Accessed: 19-Oct-2021].

4. X. Zhou, D. Wang, and P. Krähenbühl, “Objects as points,” *arXiv.org*, 25-Apr-2019. [Online]. Available: https://arxiv.org/abs/1904.07850v2. [Accessed: 19-Oct-2021].

5. Z. Wang and A. C. Bovik, “A Universal Image Quality Index,” *A universal image quality index*, Mar-2002. [Online]. Available: https://ece.uwaterloo.ca/~z70wang/publications/uqi.html. [Accessed: 25-Apr-2022].

6. M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.” [Online]. Available: https://www.cs.ait.ac.th/~mdailey/cvreadings/Fischler-RANSAC.pdf. [Accessed: 25-Apr-2022].