Augmenting Driving Image Segmentation with Lane Detection

Joshua Levine, Kshitij Phulare, Mukesh Naresh Chugani, Anant Dahiya

1. Motivation

The semantic segmentation datasets that are most popular for autonomous driving projects label a wide range of classes, including objects, people, signs, and roads. However, most of these datasets lack ground truth labels for lanes. Detecting lanes is an essential component of an autonomous system. To address this, we propose augmenting pre-trained semantic segmentation models with an analytical lane segmentation algorithm, which uses Canny Edge Detection and the Hough Transform. The proposed method can be used with any semantic segmentation module and any dataset.

2. Datasets

Due to the lack of lanes in popular autonomous driving datasets, we use two datasets, one to evaluate our lane detection approach, and another to qualitatively evaluate the composed results.

Lane detection is evaluated using the unsupervised LLAMAS dataset [3], which consists of over 100k images with splines for annotated lanes. These annotations have been captured with the help of LiDAR and hence can serve as an effective ground truth for lane detection. The dataset contains the annotation of up to four lanes in the scene, but we restrict our implementation to detecting the two nearest lanes.

We use a segmentation model trained on the Cityscapes dataset, which contains 30 classes and images from 50 cities [4]. As is often the case with these types of datasets, Cityscapes only segments the road, not individual lanes on a road. Therefore we can only qualitatively evaluate the results of applying lane detection to the Cityscapes images. The link to a subset of our datasets can be found here.

3. Approach

In order to generate segmentation masks for the 30 classes from Cityscapes and lanes, we developed a custom pipeline that feeds images to both the lane detection model and the pre-trained segmentation model.

3.1. Pipeline

We developed a versatile pipeline for feeding images into segmentation and lane detection modules, then composing the results. It was designed to work with any image dataset without modification. The segmentation method can also be replaced by changing just one function.

The design of the pipeline was dictated by the fact that the lane detection module requires a labeled road region for each image so that it does not detect all lines in each image. Consequently, we need to run semantic segmentation first to get a mask for the road, which is supplied to the lane detection module.

3.2. Segmentation

To segment each of the classes from Cityscapes, we used pre-trained Dilated Residual Networks (DRNs) [2]. In our pipeline, we use the paper's implementation of the DRN-D-22 network. We use their weights, which were trained on Cityscapes. Finally, we implemented the image processing required for segmentation ourselves.

DRNs were chosen because they perform well on the Cityscapes dataset and because their weights and architectures were available online. Dilated Residual Networks (DRNs) can address the loss of spatial acuity in image classification by increasing the resolution of output feature maps without reducing the receptive field of individual neurons. DRNs outperform non-dilated counterparts in image classification, object localization, and semantic segmentation, without increasing the model's depth or complexity.

3.3. Lane Detection

Our analytical approach to lane detection ensures that it will work on any dataset. We used Canny Edge detection, followed by Hough Transform, followed by heuristics-based lane generation. We implemented both Canny edge detection and Hough Lines from scratch. First, we smooth the image with a Gaussian filter, then find gradients with a Sobel filter. Non-Max suppression followed by gradient thresholding is applied to sharpen the detected edges and eliminate weaker gradients. A region of interest containing the road is cropped out based on the road mask generated by semantic segmentation. This cropped ROI is then passed to the Hough transform module which transforms the points to polar coordinates and then to Hough space. The intersection in Hough space is then inversely transformed to polar space and the line of best fit is found. We also threshold the minimum number of intersections to be present in a bin to qualify as a Hough line. Building on top of this, the hough

lines are heuristically filtered out to extract the most probable ones corresponding to the two lanes. Within this, lines with near to 0 or infinity slopes are eliminated. Also, to filter out the boundaries of the road, lines with x-intercept not within the image are removed. The remaining lines are partitioned into left/right based on the slopes and then their median line is calculated to avoid outliers.

4. Results

4.1. Segmentation

The pre-trained DRN network performed well on the Cityscapes validation set. We used DRN-D-22, which is one of the smaller available architectures. The reported mean IoU was 68%, which we confirmed while debugging. We provide sample outputs in Appendix A. The goal of our project was to create a pipeline for an arbitrary semantic segmentation model, so the performance of DRN was not a major concern. However, with a mean IoU of 68%, the model performs pretty well.

4.2. Lane Detection

The paper [4] which introduced the LLAMAS dataset mentions the mean absolute distance between the detected lines and the ground truth as the metric to evaluate the performance. Precisely, the mean of pixel distances between the predicted x-coordinate and the ground truth is calculated for each image. The following table shows the mean absolute distance in pixels.

	Left lane	Right Lane	Avg.
Our algorithm	0.586	1.322	0.979
MobileNet-v1 [4]	0.021	0.024	0.044

It can be seen that the performance of our implementation doesn't surpass the state-of-the-art algorithm. This is due to our method being analytical and not involving any kind of supervised learning. These coarse detections can further be used to facilitate the annotation of data or the training of other supervised models. We provide sample outputs in Appendix B.

4.3. Composed Models

Without lane labels, we are forced to evaluate our approach visually. Hence, we provide sample outputs from our pipeline in Appendix C. The first set of figures shows randomly selected outputs. The approach performed very well on these images. It can be noted that even though both sides of the lane are dashed lines, our pipeline can detect the lanes nevertheless. The second set of figures shows some failure cases. These cases arise because our lane detection is heuristic-based and does not involve any training. For instance, if the lane has a curve, our method only provides a tangent to the curve since we only detect hough lines. Additionally, if the camera placement is such that the imaginary starting point of the lanes is outside the image, then such lines would be eliminated as they would be considered boundaries of the road. Some other cases involve the presence of very less contrast between the lane lines and the road in grayscale. This would result in the lane not being detected by the edge detector.

Our pipeline processes images quickly, with an average time of 0.153s for semantic segmentation and 0.42s for lane detection.

5. Implementation Details

All code was written in Python. The GitHub repository for our implementation can be found here. We mostly used standard vision packages. The full list can be found in `requirements.txt`.

5.1. Segmentation

The DRN architecture and pre-trained weights come from Fisher Yu's Github repo [1,2]. We implement the test loop, along with image formatting, in `segment_proj.py`. To test our approach, we ran it on the full Cityscapes test set to verify the mIoU, as described in Section 4.1.

5.2. Lane Detection

For the implementation of canny and hough lines, we referred to [5,6,7] to understand the high-level concepts of the algorithms. On top of this, we figured out some of the details these sources failed to include, applied thresholding, and wrote the implementations from scratch. We evaluated our method on the LLAMAS validation set with MAD. We used a 64-core AMD ThreadRipper Pro to execute lane detection.

5.3. Pipeline

We implemented the full pipeline ourselves in `main.ipynb`. It reads all images from a directory, passes them to the semantic segmentation module then the lane detection module, and outputs the results.

6. Conclusion & Innovation

The lack of labeled lanes in autonomous driving datasets means that developers will either have to label new datasets or train multiple models in order to understand scenes well enough to control an autonomous vehicle. Our contribution is to provide a versatile zero-shot method for providing lane masks to extend the functionality of semantic segmentation models that were trained without lane labels. By using an analytical algorithm, we ensure that our approach is dataset agnostic, and can therefore be used for a wide variety of applications.

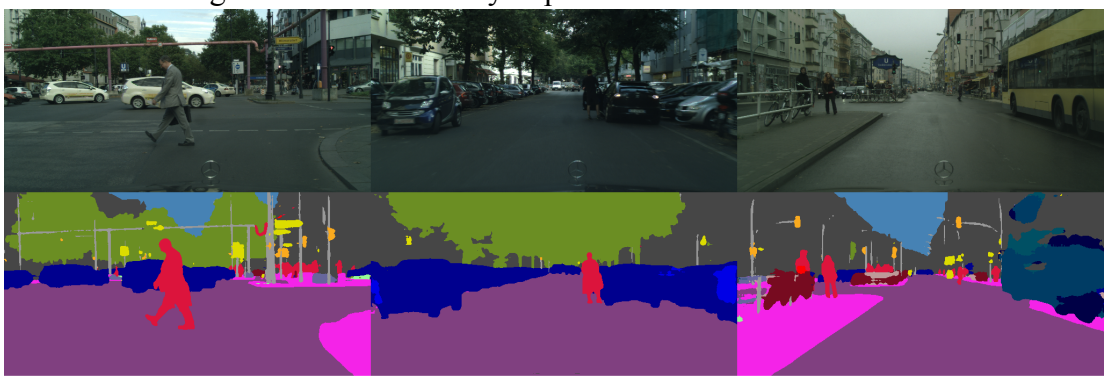
We initially attempted to detect lanes using a zero-shot learned approach, specifically CLIPSeg. Unfortunately, this method was unsuccessful in identifying any pixels as lane lines. As a result, we determined that an analytical method would be necessary for our lane detection task.

One of the challenges of our lane detection method is that it needs to know the region of the image where lane lines would be to avoid false positives. As a result, on its own, the lane detection algorithm would require manual annotations for each image or driving video. By running semantic segmentation first, we can use the mask generated for the road class to supervise the lane detection system, thereby allowing it to work on any dataset. Additionally, our essential innovation involved the use of heuristics to avoid false positives by analyzing the slopes and intercepts. Although the implementation of canny and hough lines from scratch was very challenging, it enabled us to tweak the intricacies of the algorithms (for instance, using the median to eliminate outliers).

To further visualize our pipeline's capabilities, we recorded a custom dataset in Champaign. The results of this are available in Appendix D. Note that the reflection on the car's hood altered the outputs, but beyond the hood, the approach performed well.

The complexity of our pipeline significantly increased the risk of failure, especially because the quality of the lane detection results depends on the quality of the semantic segmentation. Due to the difficulty of implementing and tuning this technique, and our efforts in curating a custom dataset, we believe we should receive 20/20 points for the innovation/challenge component of grading.

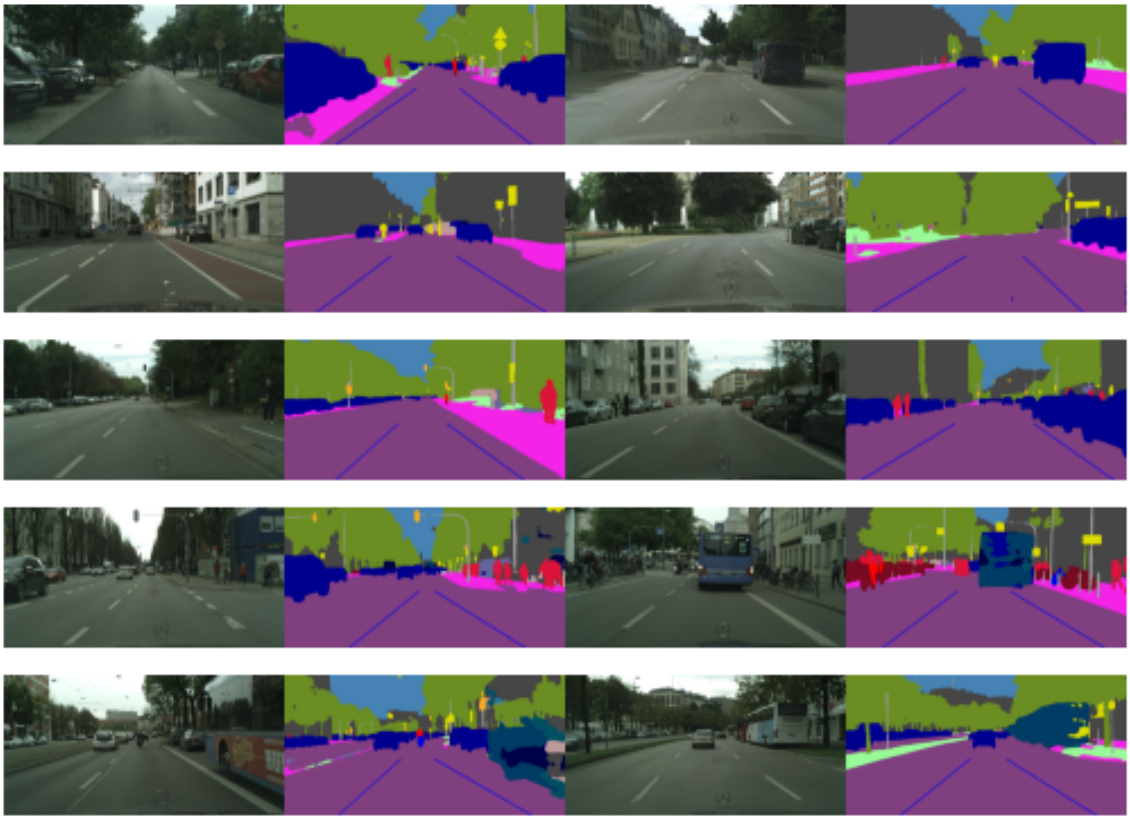
A. Semantic segmentation on the Cityscapes test set.



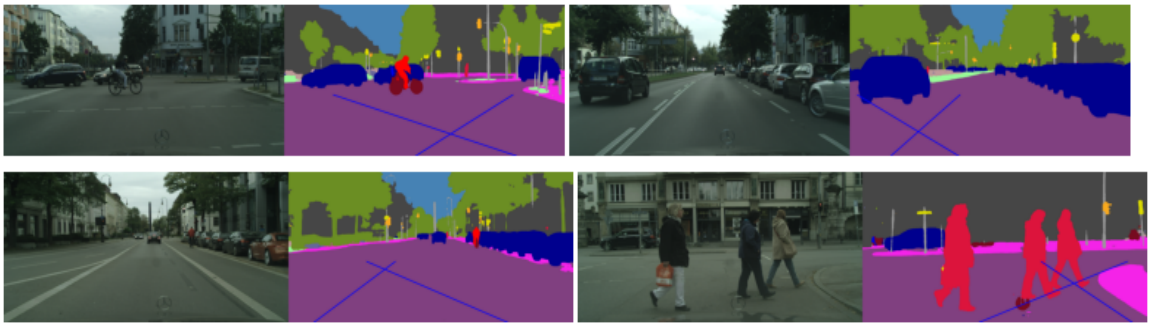
B. Lane detection Outputs

C.

Semantic Segmentation on the Cityscapes test set with Lanes



Failed Lane Detections



D. Performance on Custom Dataset (Video Link)

References

[1] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in International Conference on Learning Representations (ICLR), 2016

[2] F. Yu, V. Koltun, and T. Funkhouser, “Dilated Residual Networks,” in Computer Vision and Pattern Recognition (CVPR), 2017

[3] K. Behrendt and R. Soussan, "Unsupervised Labeled Lane Markers Using Maps," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea (South), 2019, pp. 832-839, doi: 10.1109/ICCVW.2019.00111.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

[6] https://towardsdatascience.com/a-deep-dive-into-lane-detection-with-hough-transform-8f90fdd1322f

[7] Richard O. Duda and Peter E. Hart. 1972. Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM 15, 1 (Jan. 1972), 11–15. https://doi.org/10.1145/361237.361242

[8] https://github.com/fyu/drn/tree/master