Unsupervised Labeled Lane Markers Using Maps

Karsten Behrendt
Bosch Automated Driving
karsten.behrendt@us.bosch.com

Ryan Soussan
Bosch Automated Driving
ryan.soussan@us.bosch.com

Abstract

Large and diverse annotated datasets can significantly increase the accuracy of machine learning models. However, human annotations can be cost and time intensive, and generating 3D information and connectivity for image features using manual annotations can be difficult and error prone.

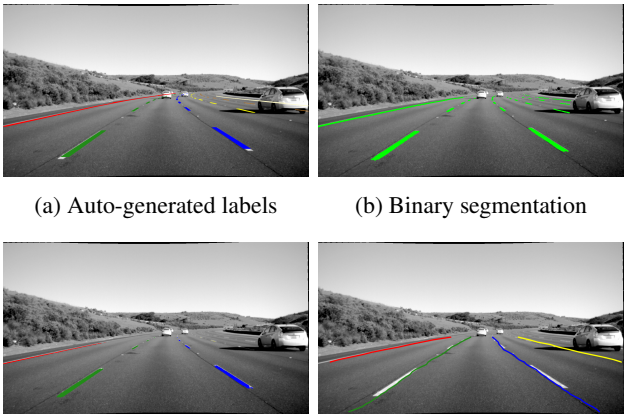
We therefore propose to automatically annotate lane markers in images and assign attributes to each marker such as 3D positions by using map data. Our method projects map lane markers into image space for far distances and relies on a sample-based optimization to refine projections and increase the accuracy of the labels.

As part of this work, we publish the Unsupervised LLAMAS dataset of 100,042 labeled lane marker images from about 350 km recorded drives which make this one of the largest high-quality lane marker datasets that is freely available. We estimate that manually annotating a dataset of this size would take several person years.

The dataset contains pixel-level annotations of dashed lane markers, 2D and 3D endpoints for each marker, and lane associations to link markers. With the dataset, we create and open source benchmark challenges for binary marker segmentation, lane-dependent pixel-level segmentation, and lane border regression to enable a straightforward comparison of different detection approaches.

1. Introduction

Over the last few years, deep learning has significantly increased the accuracy of classification [1], localization, segmentation [2], and detection [3] tasks for computer vision. To reach the full potential of deep neural networks and achieve the best generalization, large and diverse datasets are needed. Some of the largest annotated datasets are the ImageNet collection [1] and the COCO dataset [4] which have led to many improvements in the field. Both of these datasets were annotated by hand, which can be very challenging, time-consuming, and labor intensive.



(a) Auto-generated labels (b) Binary segmentation

(c) Multi-class segmentation (d) Spline regression

Figure 1: Image 1a shows a labeled lane image with lane associations produced by our automated labeler pipeline. Images 1b-1d show three models trained to output different lane marker representations using our auto-generated dataset.

Automatically generating datasets with limited human input can lead to faster research and development cycles for the computer vision community. Krasin, Duerig, and Alldrin *et al*. [5] created the OpenImages dataset by using a highly-accurate model to predict objects in images. They then relied on an additional manually annotated validation set to reduce the number of false positives. Another popular approach for creating large labeled datasets without using human annotations is to use simulation [6]. A promising technique is training neural networks to post-process simulated images that more closely resemble reality [7–9]. The resulting images are visually appealing, but may contain noticeable differences compared with real images. Simulated datasets often need to be extended with real annotated camera images, as is the case for automated driving.

In this work, we present the Unsupervised LLAMAS dataset, a lane marker dataset of 100,042 images with labeled lane markers. We automatically annotate lane makers in images using highly-accurate maps and an additional op-

timization step to improve labeling accuracy. Human effort is limited to collecting lidar and camera data for the map generation pipeline and finally removing faultily labeled images from the labeling procedure, minimizing hands-on labeling efforts. Our approach successfully labels 88% of images, with the majority of faultily labeled images being pruned due to errors in the mapping procedure.

With the Unsupervised LLAMAS dataset, we create and open source benchmark challenges on binary marker segmentation, multiclass segmentation, and lane regression.

2. Related Work

The absence of large publicly available lane marker datasets makes comparing existing methods difficult, to the point where often only limited quantitative analysis is performed [10–14]. Other works rely on private datasets for evaluation [15–18].

There are different lane marker representations in a few existing datasets. The Road Marking dataset, for example, uses 2D boxes for lane marker annotations. Caltech Lanes [19] and the CULane dataset [**?**] represent lanes with 2D splines. BDD100k [**?**] uses 2D lines in a large amount of images but does not offer an evaluation method so far. The Camvid [**?**] and KITTI ROAD [**?**] datasets offer pixel level annotations, but are fairly small. For an overview of different datasets sizes and marker representations, we refer to Table 1.

One large dataset of labeled lane splines in images was created by [16] by projecting high intensity LIDAR points from maps, but the resulting labels needed manual corrections. A similar approach was performed by [15] where a map is represented as curves and projected into the image. Neither of these datasets are publicly available and have accuracies limited to 80 meters without containing dashes for lane markers.

The Unsupervised LLAMAS dataset is automatically annotated with high accuracy and contains labels up to 120 meters. A unique feature of our dataset is the variety of information provided with 2D and 3D lines, individual dashed markers, pixel level segmentation, and lane associations.

3. Dataset Generation

We used 14 highway recordings of around 25 km each to gather sensor data for labeling images in our dataset. Our automated labeling pipeline creates annotations for lane markers using highly accurate maps. It requires groundtruth poses for the vehicle with respect to the map, which can be provided by using sensor data recorded during mapping or by localizing a new run against map data. Each step is done automatically using either our mapping procedure or an offline variant of localization, as described in section 3.1. After projecting map markers into images, we further optimize the projections using image space detections of markers, which we describe in section 3.2. This yields the high accuracy that our dataset provides. Additionally, we create an algorithm to generate lane associations for each marker projected into an image, as our lane marker maps do not contain lane associations, which is detailed in section 3.3.

3.1. Generating Labeled Data using Maps

In this section, we describe our automated labeling pipeline used to generate labeled lane marker images from our maps. We use the following notation for frames and transforms throughout this paper: ${}^{\mathbf{B}}_{\mathbf{A}}\mathbf{T}$ denotes the rigid body transform from frame **A** to **B** $\in \mathrm{SE}(3)$ [24], where frame **A** describes the space $\in \mathbb{R}^3$ whose origin is at the position of **A**.

Our mapping pipeline automatically creates highly accurate maps for localization, including dashed lane markers in 3D, lidar intensity maps, and radar maps by fusing LIDAR, radar, camera, odometry, and GPS data together from several sensor recordings taken over the same area. We use a GraphSLAM (simultaneous localization and mapping) approach similar to that used by Levinson and Thrun [23] to generate these maps from the recorded data which we then use during online localization for the automated vehicle.

The automated labeler pipeline in Figure 2 combines our maps with sensor readings to create labeled data. Part A represents the sensor recordings, containing sensor data and transforms ${}^{\mathbf{S}}_{\mathbf{V}}\mathbf{T}$ for each sensor, where **S** is the sensor frame and **V** is the vehicle frame. The vehicle frame is the center of the rear axle as this is preferred for state estimation for odometry. The sensor transforms are generally dynamic, as most sensors are mounted on the vehicle body which moves with respect to the rear axle when driving, and therefore they are estimated throughout sensor recordings and each timestamp contains a different set of sensor transforms.

Part B shows the sources for localization estimates ${}^{\mathbf{V}}_{\mathbf{M}}\mathbf{T}$ where **M** is the global map frame. There are two different methods of obtaining ${}^{\mathbf{V}}_{\mathbf{M}}\mathbf{T}$: (1) a highly accurate offline version of our localization algorithm and (2) using localization estimates from our mapping pipeline.

The offline version of our localization algorithm uses more computational resources and denser sensor data than the online version to generate a more accurate pose estimate. It uses different combinations of sensor and map data to generate corrections for the vehicle's position, then combines these estimates into a final pose.

Since our mapping pipeline generates localization estimates using SLAM, we can leverage these poses as groundtruth for projecting our map lane markers into image space. When using this option the automated labeler is limited to using the subset of recordings used for mapping, whereas the offline localization algorithm works for any recordings created in areas where we have map data but

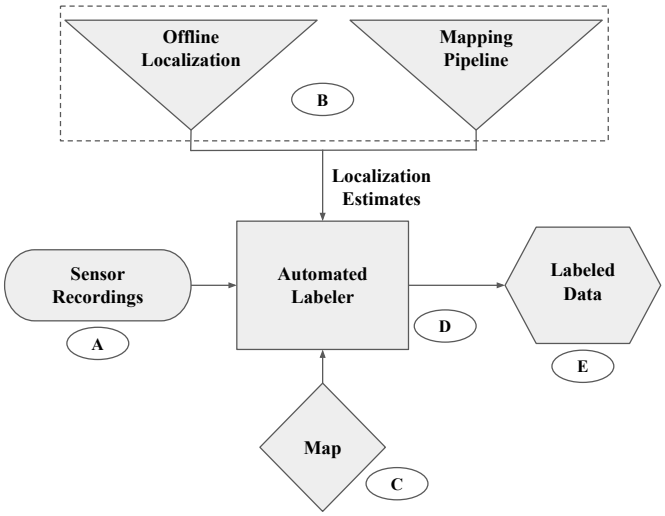
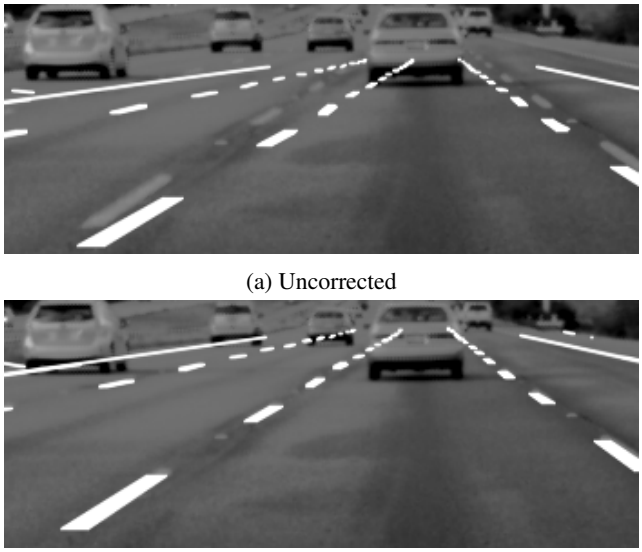


Figure 2: Automated labeler pipeline



(a) Uncorrected

(b) Corrected

Figure 3: Magnified comparison of lane projections before and after applying corrections to the camera pose. Pitch errors are exaggerated for visibility in Figure 3a.

produces slightly less accurate localization estimates.

To combine map data with sensor data, the automated labeler uses localization estimates from Part B to find the transform bringing the map data from global coordinates to vehicle coordinates (${}^{V}_{M}\mathbf{T}$). It then uses ${}^{S}_{V}\mathbf{T}$ from Part A to transform the data from vehicle coordinates to sensor coordinates. Then, for each 3D map element $\mathbf{e_m}$ in global coordinates, it generates an element in sensor coordinates $\mathbf{e_s}$ using (1).

$$\mathbf{e_s} = {}^{S}_{V}\mathbf{T}\,{}^{V}_{M}\mathbf{T}\,\mathbf{e_m} \tag{1}$$

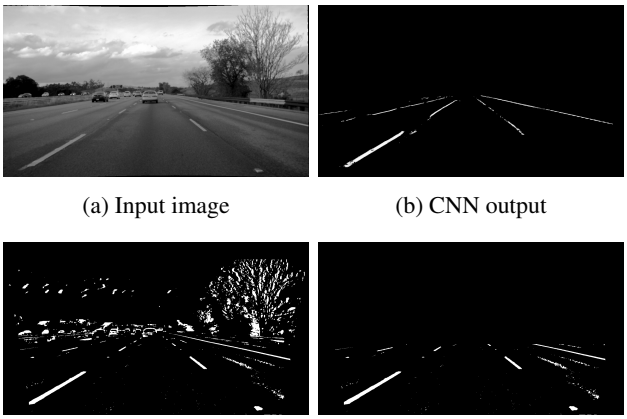
$\mathbf{e_m}$ can be any type of map data, for example the 3D endpoints of lane markers or point cloud data. Once the map data is in sensor coordinates, it can be combined with sensor readings to generate labeled data. When using camera images as sensor data, the camera calibration matrix is then used to project map elements from the sensor frame to 2D image space.

3.2. Correcting Lane Marker Map Projections

Small translational errors in ${}^{V}_{M}\mathbf{T}$ or ${}^{S}_{V}\mathbf{T}$ on the order of centimeters can lead to misaligned map data for an entire image, and errors in roll, pitch, and yaw can scale and deform map data as shown in Figure 3a. An error of one degree in pitch for example can cause markers 80 meters away to be offset by roughly 1.4 meters. Therefore, we develop a correction algorithm to fine-tune the lane map projections used by the automated labeler pipeline described in Section 3.1. We first estimate a correction ${}^{F}_{C}\mathbf{T}$, where $\mathbf{F}$ is the corrected camera frame, to the camera to vehicle transform ${}^{C}_{V}\mathbf{T}$ by viewing a series of images from a recording and manually testing different corrections to improve the map projection. We estimate a correction using the camera frame since the camera is mounted on the car body, which enables us to intuitively correct x, y, z, roll, pitch, and yaw errors. Additionally, ${}^{C}_{V}\mathbf{T}$ contains the highest amount of error of the transforms needed for the labeling pipeline. Since we used the same vehicle and sensor setup for all of our recordings, we were able to apply this estimate to other recordings as well without having to repeat this procedure.

For each image output by our labeler pipeline, the algorithm samples 100,000 correction hypotheses ${}^{F}_{C}\mathbf{T}_i$ using ${}^{F}_{C}\mathbf{T}$. We rely on a graph optimization procedure when matching map elements with image space detections during online localization for performance reasons, but in our offline labeling pipeline we can produce better labels by relying on this slower but more accurate sample-based algorithm. The algorithm then creates a labeled lane image I_{p_i} for each sample after applying the correction ${}^{F}_{C}\mathbf{T}_i$ to the map data using (1). It finally tests each I_{p_i} using the image heuristic described in Equation 2 to find the best correction.

The heuristic in (2) compares I_{p_i} with an estimated lane image I_e. We initially create I_e based on a simple top-hat filter that roughly finds light patches in the image as shown in Figure 4c. Since this filter only works for close markers, we disregard detections in the upper part of the image, see Figure 4d. Once we have enough images to train a CNN, we can improve the corrections by using the trained model's output for I_e. In theory, this approach can be run iteratively, improving the corrections with increasingly accurate segmentation models.



(a) Input image (b) CNN output

(c) Top-hat filter image (d) Lower top-hat image

Figure 4: Estimate images I_e based on CNN or initially, a simple top-hat filter heuristic. A trained segmentation model allows for more accurate corrections, but even the simple filter significantly reduces projection errors

$$s(u,v) = \begin{cases} 2 & \text{if } I_e(u,v) = I_{p_i}(u,v) = 1 \\ -1 & \text{if } I_e(u,v) = 1, I_{p_i}(u,v) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The heuristic in (2) compares labels for each pixel u, v in I_e and I_{p_i}. The first case rewards agreement for positive lane labels in both the images whereas the second penalizes the case where I_e contains a positive label that I_{p_i} does not. This was done since I_e contained highly confident positive lane labels as previously described, and thus the heuristic encourages overlap between these labels and map markings. Positive labels in I_{p_i} that are not present in I_e are not penalized since I_{p_i} contains many more labels from further distances and wider ranges than I_e.

The output of running this procedure for each image in a recording is a set of corrected labeled images I_{p_c} that are then used for our dataset. Figure 3b illustrates the improvement in the map projection after fine-tuning the correction with this algorithm.

3.3. Generating Lane Associations

To generate lane associations for each marker, we use their 3D endpoints contained in our maps.

The algorithm first transforms the lane markers to the 3D camera frame using (1). It then creates lanes by first finding the best matching markers (if they exist) in front of and behind each marker with respect to the camera. Matching markers are grouped into lanes and ordered based on their position with respect to the camera. Lanes to the left of the camera are labeled l_0, l_1, and so on, where l_0 is the closest left lane to the camera, and likewise for lanes to the right.

Dataset	# Images	Resolution	Labels
BDD100k [?]	100,000	1280x720	2D lines, no benchmark
Caltech Lanes [19]	1,224	640x480	lanes as 2D splines
Camvid [?]	701	920x720	pixel level
CULane Dataset [?]	133,235	1640x590	lanes as 2D splines
KITTI Road [?]	579	1392x512	pixel level, not markings
Road Marking [?]	1443	800x600	2D boxes
VPGNet [?]	21,097	1288x728	2D Splines, not public
Ours	**100,042**	**1280x717**	**2D, 3D dashed lines pixel level lane associations**

Table 1: Overview of lane marker datasets.

Two criteria make a candidate lane marker a marker's best match: the distance from the center of the candidate to the lane marker is smaller than 1 meter, or its score using the matching heuristic described in (3) is larger than any previous match. The first criteria forces lane markers that are very close or even overlapping (possible due to map errors) to match. The second filters out lane markers that are either too far apart or whose orientations are not well aligned.

$$h = a|\mathbf{d}|^{-1} \quad (3)$$

Here $\mathbf{d}$ is the difference between marker endpoints, using the start point of the marker longitudinally further from the camera and the endpoint of the other. The value a is the alignment score, calculated using:

$$a = \hat{\mathbf{d}}^T \cdot \hat{\mathbf{l}} \quad (4)$$

where $\hat{\mathbf{l}}$ is a unit vector oriented along the direction of the lane marker and $\hat{\mathbf{d}}$ is the unit vector for $\mathbf{d}$. The dot product in (4) yields low alignment values for markers in adjacent lanes and large ones for markers that are longitudinally aligned and therefore likely in the same lane.

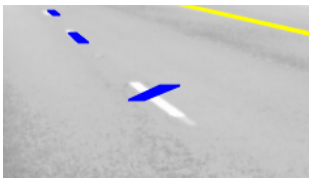
The algorithm uses California highway lane spacing requirements as a last filter to discard potential matches. Since highway markers in the same lane in California are separated by at most 10.98 meters [25], if the distance between the candidate and lane marker exceeds 11 meters the candidate is not added as a match. Additionally, if the alignment of the markers is less than 0.95, the candidate is also discarded. The alignment minimum was determined empirically.

4. Dataset Evaluation

The Unsupervised LLAMAS dataset is one of the largest lane public lane marker datasets. It offers a variety of marker representations that regularly outperforms existing datasets as shown in Table 1. The combination of 2D, 3D lines, projected pixel level annotations, lane associations,

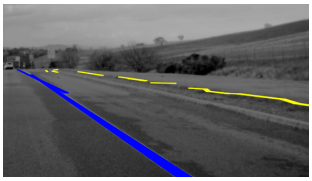
and calculated 2D splines enable the training and evaluation of a variety of different approaches. With the public benchmark challenges, it has the potential to further research in lane marker detections and semantic segmentation specific to the properties of markers. For an overall impression of our dataset and the trained models, we refer the reader to our supplementary video.

4.1. Dataset and Projection Quality

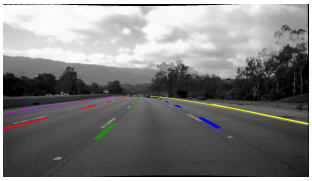


(a) Map marker alignment error

(b) Occluded underpass



(c) False lane in map



(d) Bad projection

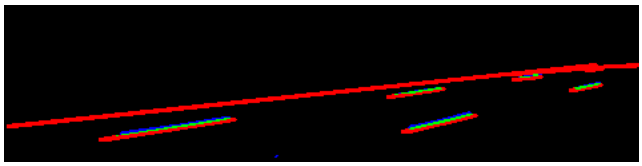
Figure 5: Fatal errors during dataset generation

Our automatically generated dataset generally offers high accuracy with parts being accurate up to more than 100 meters. Having map information allows us to regularly annotate markers further than annotators would be able to confidently do so. The very small number of pixels at greater distances make lane markers not only harder to annotate but also harder to detect by trained models.

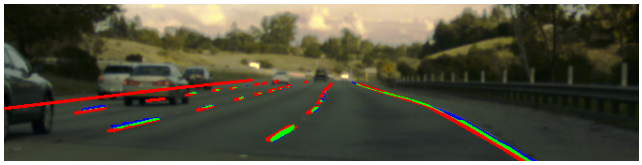
We visually inspect the images generated by our labeling pipeline and filtered out any images containing fatal errors. Fatal errors are defined as samples with severe misalignment in the map or image, containing faulty map elements, or missing lane markers due to map errors. Our approach does not account for 3D occlusions such as underpasses, bridges, or dynamic objects. Approximately 12% of images contained fatal errors, primarily present in ordered sequences of images containing map errors. Some of the fatal errors are displayed in Figure 5.

Labels that mostly overlap with their respective markings and showed no damaging orientation or translation errors were kept. The final dataset does contain some inaccuracies though with examples shown in Figure 7. There are minor flaws in the map which were not removed such as closely overlapping markers. Additionally, the lane marker widths in the used map are not very accurate which is why we set a fixed with of 12 cm for all markers.

(a) Image to annotate

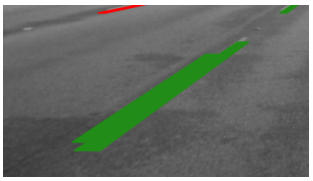


(b) Manual and automated annotations for Figure 6a

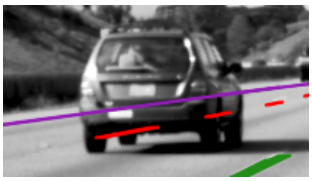


(c) Distant markers are regularly missed within manual annotations. For visibility, they are drawn without anti aliasing and are still only two pixels wide in the original image size.

Figure 6: Comparison of manual (blue) and automated (red) annotations. Pixels with manual and automated annotations are drawn in green.



(a) Double lane markers



(b) Ignoring occlusion

Figure 7: Inaccuracies in the dataset

LLAMAS only	human only	intersection	negative
0.75%	0.36%	0.90%	97.98%

Table 2: Overlap of automatic and human annotations.

Lane markers are projected into the image for some distances where they can no longer be clearly identified by human annotators anymore. Figure 6 shows example cases where annotations are created at far distances and contain a lane boundary that we missed to annotate in our experiments. Table 2 gives an impression of the differences between human and automatic annotations. Overall, only about two percent of all pixels are annotated as markers. The additional automatically annotated pixels are a combination of markers on dynamic objects, far out annotations, and additionally marked pixels with only little overlap in the projection. Given the 3D information of pixels, it is easily

possible to remove pixel level annotations beyond a fixed distance if that is advantageous.

During our experiments, we on average needed about 9 minutes to manually annotate a single image. To annotate the complete dataset, we would need about 375 40-hour weeks. Even if the annotation time can be reduced, automatic annotations still have the potential to drastically reduce development times and cost.

5. Benchmark Challenges and Baselines

Depending on the application, different marker representations are useful for driver assistance and automated driving systems, and there exist a variety of different modeling options for lane markers [10, 14, 16, 18, 19].

We focus on pixel-level segmentation and regressed lane borders which are represented by curves, such as splines. Pixel-level segmentation can be very accurate but is computationally expensive to create and use, whereas curves are faster to create and provide the user with a higher level representation of lanes.

For this, we provide a pre-defined dataset split into 58,269 training, 20,844 validation, and 20,929 test samples. We open source the benchmarks, evaluation scripts, and website, so that adding additional metrics, and challenges is possible. In the following sections, we cover the initial benchmark challenges.

5.1. Binary Pixel-level Segmentation

Pixel level segmentation is useful for automated driving as it enables the comparison of image data to map data and the acquisition of semantic information in the current environment. It can be used for localization, mapping, and environment sensing purposes. The binary segmentation problem is scored based on average precision (AP) on a per pixel basis. One challenging aspect of this dataset is the low number of positive pixels which account for less than 2% of the image.

As our baseline method for detecting lane marker pixels, we train a DeepLabv3+ [26] model with Xception [27] backbone. Overall, it achieves an AP of 38.0% at an inference time of about 200 ms in native Tensorflow [**?**] on an Nvidia GeForce GTX 1080 Ti.

5.2. Pixel-level Segmentation with Lane Association

In addition to only detecting marker pixels, some applications such as localization may benefit from lane associations. This multiclass segmentation problem is evaluated based on mean average precision (mAP).

As a baseline, we again train the DeepLabV3+ [26] but also classify a marker's associated lane. For simplicity, we limit the lane associations to the neighboring two lanes, l_1, l_0, r_0, and r_1 as described in Section 3.3. Overall, we achieve a mean average precision of 31.2%. The closer markers of lane borders l_1 and l_2 are detected at significantly higher accuracy which is shown in Table 4.

mAP	l_1	l_0	r_0	r_1
31.2%	17.2%	40.9%	41.3%	17.5%

Table 3: Average precisions of the multi-class marker segmentation problem on the test set.

5.3. Lane Border Regression

To generate smooth trajectories, driver assistance and automated driving systems benefit from knowledge of lane structure. The farther lanes can be detected, the more these systems can plan ahead even without map information. In theory, lane borders can even be estimated farther than there are visible lane markers by looking at other traffic participants and the general road structure. We aim to accurately detect the lane borders and associate them with the appropriate lanes in the image space.

As a first step, we create splines that describe the lane borders for the existing labels. To limit oscillations between the control points, we uniformly interpolate additional points between the markers start and endpoints. The benchmark again focuses on the four closest lane borders l_1, l_0, r_0, and r_1, as described in Section 3.3.

$$L_{LB} = \frac{\sum_{l_j \in l_1, l_0, r_0, r_1} \sum_i^n t_{e_{li}} |t_{x_{l_j i}} - p_{x_{l_j i}}|}{4n} \quad (5)$$

It is scored on the mean horizontal distance between the estimated and actual lane border for each pixel i along the y-axis as in Equation 5. The network prediction value $p_{x_{li}}$ estimates target values $t_{x_{li}}$ along the x-axis for a given lane l_j and step i. $t_{e_{l_j i}}$ is set to 1 if a target x coordinate exists for a given vertical pixel i for a given lane l_j. Extensions could also take the absolute distance in 3D coordinates, missing lane borders, or different weightings into account.

As a base architecture, we implement a MobileNet v1 [28] with 200 output units that represent 50 pixels along the y-axis in the image for each lane. The model aims to closely approximate the label splines by regressing uniformly sampled points vertically along the image. In (6), the regression loss L_R to minimize is defined as a weighted L2 loss for all points where a target is specified, *i.e.*, if $t_{e_{li}} = 1$.

$$R_L = \sum_{l_j \in \{l_1, l_0, r_0, r_1\}} \sum_i^{n=50} t_{e_{li}} (p_{x_{li}} - t_{x_{li}})^2 \quad (6)$$

For our baseline model, the mean deviation for each point between the predicted and target curve is approximately 14.8 pixels. As expected, the accuracy for the closer lane borders is again higher than for the outer ones. The results for each lane border are displayed in Table 4.

Mean Deviation	l_1	l_0	r_0	r_1
14.8	16.8	11.0	12.5	20.6

Table 4: Mean deviation in pixels between predicted and annotated curve in x-axis.

6. Conclusion

In this paper, we presented the Unsupervised LLAMAS dataset consisting of dashed lane markers with pixel-level annotations, endpoints of individual markers in 3D and image space, and lane associations for each marker. Our pipeline for generating the labeled images utilizes automatically created maps to project markers into camera images and relies on a further optimization procedure to increase the accuracy of the labels. We use this method to create the largest annotated lane marker dataset with over 100,000 annotated images. A manually annotated dataset of this size would likely take multiple person years to annotated.

With the dataset, we publish a benchmark and baseline methods for lane marker detection. The challenges presented are a pixel-level binary segmentation, a segmentation problem with lane association, and a lane estimation task. For a qualitative impression of the dataset and baseline methods, we encourage the reader to view the supplementary video. We look forward to new approaches, metrics, and challenges based on this dataset.

Future work may focus on objects beyond lane markers, even larger datasets, different sensors, urban environments, different weather conditions, and optimizing segmentation methods for lane marker segmentation.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 2012, pp. 1106–1114. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks 1

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. 1

[3] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016. 1

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755. 1

[5] I. Krasin, T. Duerig, N. Alldrin, A. Veit, S. Abu-El-Haija, S. Belongie, D. Cai, Z. Feng, V. Ferrari, V. Gomes *et al.*, "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github.com/openimages*, vol. 3, 2016. 1

[6] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision*. Springer, 2016, pp. 102–118. 1

[7] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *CoRR*, vol. abs/1612.07828, 2016. [Online]. Available: http://arxiv.org/abs/1612.07828 1

[8] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," *CoRR*, vol. abs/1707.09405, 2017. 1

[9] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016. [Online]. Available: http://arxiv.org/abs/1611.07004 1

[10] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai, "Multi-channel lidar processing for lane detection and estimation," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, pp. 1–6. 2, 6

[11] T. Kasai and K. Onoguchi, "Lane detection system for vehicle platooning using multi-information map," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 1350–1356. 2

[12] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, "A novel lane detection based on geometrical model and gabor filter," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 59–64. 2

[13] L.-W. Tsai, J.-W. Hsieh, C.-H. Chuang, and K.-C. Fan, "Lane detection using directional random walks," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 303–306. 2

[14] S. Kammel and B. Pitzer, "Lidar-based lane marker detection and mapping," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 1137–1142. 2, 6

[15] K. Behrendt and J. Witt, "Deep learning lane marker segmentation from automatically generated labels," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017. 2

[16] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving," *CoRR*, vol. abs/1504.01716, 2015. [Online]. Available: http://arxiv.org/abs/1504.01716 2, 6

[17] R. F. Berriel, E. de Aguiar, V. V. de Souza Filho, and T. Oliveira-Santos, "A particle filter-based lane marker tracking approach using a cubic spline model," in *2015 28th*

SIBGRAPI Conference on Graphics, Patterns and Images. IEEE, 2015, pp. 149–156. 2

[18] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, no. 4, pp. 269–280, 2004. 2, 6

[19] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 7–12. 2, 4, 6

[20] tuSimple, "Lane detection challenge," May 2017. [Online]. Available: http://benchmark.tusimple.ai/#/t/1

[21] 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," *CoRR*, vol. abs/1604.01685, 2016. [Online]. Available: http://arxiv.org/abs/1604.01685

[22] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[23] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4372–4378. 2

[24] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012, vol. 26. 2

[25] Department of Transportation, "Traffic manual chapter 6 - markings," State of California, Tech. Rep., Jul. 1996. 4

[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018. 6

[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1800–1807. 6

[28] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. 6