A Dataset for Lane Instance Segmentation in Urban Environments

Brook Roberts, Sebastian Kaltwang, Sina Samangooei,
Mark Pender-Bare, Konstantinos Tertikas, and John Redford

FiveAI Ltd., Cambridge CB2 1NS, U.K.
{brook,sebastian,sina,mark.pender-bare,konstantinos,john}@five.ai

Abstract. Autonomous vehicles require knowledge of the surrounding road layout, which can be predicted by state-of-the-art CNNs. This work addresses the current lack of data for determining lane instances, which are needed for various driving manoeuvres. The main issue is the time-consuming manual labelling process, typically applied per image. We notice that driving the car is itself a form of annotation. Therefore, we propose a semi-automated method that allows for efficient labelling of image sequences by utilising an estimated road plane in 3D based on where the car has driven and projecting labels from this plane into all images of the sequence. The average labelling time per image is reduced to 5 seconds and only an inexpensive dash-cam is required for data capture. We are releasing a dataset of 24,000 images and additionally show experimental semantic segmentation and instance segmentation results.

Keywords: dataset · urban driving · road · lane · instance segmentation · semi-automated annotation · partial labels

1 Introduction

Autonomous vehicles have the potential to revolutionise urban transport. Mobility will be safer, always available, more reliable and provided at a lower cost. Yet we are still at the beginning of implementing fully autonomous systems, with many unsolved challenges remaining [1]. One important problem is giving the autonomous system knowledge about surrounding space: a self-driving car needs to know the road layout around it in order to make informed driving decisions. In this work, we address the problem of detecting driving lane instances from a camera mounted on a vehicle. Separate, space-confined lane instance regions are needed to perform various challenging driving manoeuvres, including lane changing, overtaking and junction crossing.

Typical state-of-the-art CNN models need large amounts of labelled data to detect lane instances reliably (e.g. [2,3,4]). However, few labelled datasets are publicly available, mainly due to the time consuming annotation process; it takes from several minutes up to more than one hour per image [5,6,7] to annotate images completely for semantic segmentation tasks. In this work, we introduce a new video dataset for road segmentation, ego lane segmentation and lane

instance segmentation in urban environments. We propose a semi-automated annotation process, that reduces the average time per image to the order of seconds. This speed-up is achieved by (1) noticing that driving the car is itself a form of annotation and that cars mostly travel along lanes, (2) propagating manual label adjustments from a single view to all images of the sequence and (3) accepting non-labelled parts in ambiguous situations.

Previous lane detection work has focused on detecting the components of lane boundaries, and then applying clustering to identify the boundary as a whole [8,9,10,2]. More recent methods use CNN based segmentation [2,4], and RNNs [11] for detecting lane boundaries. However, visible lane boundaries can be interrupted by occlusion or worn markings, and by themselves are not associated with a specific lane instance. Hence, we target lane instance labels in our dataset, which provide a consistent definition of the lane surface (from which lane boundaries can be derived). Some work focuses on the road markings [12], which are usually present at the border of lanes. However, additional steps are needed to determine the area per lane. Much of the work has only been evaluated on proprietary datasets and only few public datasets are available [13]. Various

Fig. 1. Example image from our dataset (top left), including annotations for road (top right), ego-lane (bottom left) and lane instance (bottom right). Road and lanes below vehicles are annotated despite being occluded. Non-coloured parts have not been annotated, i.e. the class is not known.

datasets include road area as a detection task, in addition to many other semantic segmentation classes [14,5,15,16,6,7,17]. Some datasets also includes the ego-lane [18], which is useful for lane following tasks. Few datasets provide lane instances [19,20], which are needed for more sophisticated driving manoeuvres. Aly et. al. [19] provide a relatively limited annotation of 4 single coordinates per lane border. TuSimple [20] offer a large number of sequences, but for highway driving only. Tab. 1 provides an overview of the publicly available datasets. Our

average annotation time per image is much lower. However, our provided classes are different, since we focus on lane instances (and thus ignore other semantic segmentation classes like vehicle, building, person, etc.). Furthermore, our data provides road surface annotations in dense traffic scenarios despite occlusions, i.e. we provide the road label below the vehicles (see Fig. 1). This is different from typical semantic segmentation labels, which provide a label for the occluding object instead [14,5,15,16,6,7]. Another approach to efficiently obtain labels is to create a virtual world where everything is known a-priori [21,22,23]. However, current methods do not reach the fidelity of real images.

Table 1. Comparison of the available datasets. Label time per image is only shown if provided by the authors. Many datasets are not only targeting the road layout, and thus the labelling includes more classes.

Name	Year	#labeled frames	#videos	img. seq.	road area	ego lane	lane instances	label time per img.
Caltech Lanes [19]	2008	1,224	4	✓	✓[b]	-	✓	-
CamVid [14,5]	2008	701	4	✓	✓	-	-	20 min
Yotta [15]	2012	86	1	✓	✓	-	-	-
Daimler USD [16]	2013	500	-	-	✓[c]	-	-	-
KITTI-Road [18]	2013	600	-	-	✓	✓	-	-
NYC3DCars [17]	2013	1,287	-	-	✓	-	-	-
Cityscapes [6] (fine)	2016	5,000	-	✓[a]	✓	-	-	90 min
Cityscapes [6] (coarse)	2016	20,000	-	✓[a]	✓	-	-	7 min
Mapillary Vistas [7]	2017	20,000	-	-	✓	-	-	94 min
TuSimple [20]	2017	3,626	3,626	✓[a]	✓[b]	✓	✓[d]	-
Our Lanes	2018	23,980	402	✓	✓	✓	✓	5 sec

[a] Only single images are annotated, but additional (non-annotated) image sequences are provided.

[b] Road area is implicitly annotated by the given lanes.

[c] Annotated ground instead of road, i.e. it includes non-drivable area.

[d] Limited to three instances: ego-lane and left/right of ego-lane.

Some previous work has aimed at creating semi-automated object detections in autonomous driving scenarios. [24,17] use structure-from-motion (SFM) to estimate the scene geometry and dynamic objects. [25] proposes to annotate lanes in the birds-eye view and then back-project and interpolate the lane boundaries into the sequence of original camera images. [26] uses alignment with Open-StreetMap to generate ground-truth for the road. [27] allows for bounding box annotations of Lidar point-clouds in 3D for road and other static scene components. These annotations are then back-projected to each camera image as semantic labels and they report a similar annotation speed-up as ours: 13.5 sec per image. [28] propose to detect and project the future driven path in images, without the focus of lane annotations. This means the path is not adapted to lane widths and crosses over lanes and junctions. Both [27,28] require an expensive sensor suite, which includes calibrated cameras and Lidar. In contrast, our method is applicable to data from a GPS enabled dash-cam. The overall contributions of this work include: (1) The release of a new dataset for lane instance and road segmentation, (2) A semi-automated annotation method for lane in-

stances in 3D, requiring only inexpensive dash-cam equipment, (3) Road surface annotations in dense traffic scenarios despite occlusion, and (4) Experimental results for road, ego-lane and lane instance segmentation using a CNN.

2   Video Collection

Videos and associated GPS data were captured with a standard Nextbase 402G dashcam recording at a resolution of 1920x1080 at 30 frames per second and compressed with the H.264 standard. The camera was mounted on the inside of the car windscreen, roughly along the centre line of the vehicle and approximately aligned with the axis of motion. Fig. 1 (top left) shows an example image from our collected data. In order to remove parts where the car moves very slowly or stands still (which is common in urban environments), we only include frames that are at least 1m apart according to the GPS. Finally, we split the recorded data into sequences of 200m in length, since smaller sequences are easier to handle (e.g. no need for key-frame bundle adjustment, and faster loading times).

3   Video Annotation

The initial annotation step is automated and provides an estimate of the road surface in 3D space, along with an estimate for the ego-lane (see Sec. 3.1). Then the estimates are corrected manually and further annotations are added in the road surface space. The labels are then projected into the 2D camera views, allowing the annotation of all images in the sequence at once (see Sec. 3.2).

3.1   Automated Ego-lane Estimation in 3D

Given a dash-cam video sequence of N frames from a camera with unknown intrinsic and extrinsic parameters, the goal is to determine the road surface in 3D and project an estimate of the ego-lane onto this surface. To this end, we first apply OpenSfM [29], a structure from motion algorithm, to obtain the 3D camera locations $\mathbf{c}_i$ and poses $\mathbf{R}_i$ for each frame $i \in \{1, ..., N\}$ in a global coordinate system, as well as the camera projective transform $P(\cdot)$, which includes the estimated focal length and distortion parameters ($\mathbf{R}_i \in \mathbb{R}^{3\times3}$ are 3D rotation matrices). OpenSfM reconstructions are not perfect, and failure cases are filtered during the manual annotation process.

We assume that the road is a 2D manifold embedded in the 3D world. The local curvature of the road is low, and thus the orientation of the vehicle wheels provide a good estimate of the local surface gradient. The camera is fixed within the vehicle with a static translation and rotation from the current road plane (i.e. we assume the vehicle body follows the road plane and neglect suspension movement). Thus the ground point $\mathbf{g}_i$ on the road below the camera at frame i is calculated as $\mathbf{g}_i = \mathbf{c}_i + h\mathbf{R}_i\mathbf{n}$, where h is the height of the camera above the road and $\mathbf{n}$ is the surface normal of the road relative to the camera (see Fig. 2, left). The left and right ego-lane borders $\mathbf{b}_i^{left}, \mathbf{b}_i^{right}$ can then be derived as

Fig. 2. Estimation of the lane border points $\mathbf{b}_i^{left}, \mathbf{b}_i^{right}$ at frame i. $\mathbf{c}_i$ is the camera position at frame i (obtained via SfM), $\mathbf{g}_i$ is point on the road below the camera, h is the height of the camera above the road, $\mathbf{f}$ is the forward direction, $\mathbf{n}$ is the normal vector of the road plane, $\mathbf{r}$ is the horizontal vector across the lane ($\mathbf{f}$, $\mathbf{n}$ and $\mathbf{r}$ are relative to the camera orientation) and w_i^{left}, w_i^{right} are the distances to the left and right ego-lane borders.

$$\mathbf{b}_i^{left} = \mathbf{g}_i + w_i^{left}\mathbf{R}_i\mathbf{r}$$
$$\mathbf{b}_i^{right} = \mathbf{g}_i + w_i^{right}\mathbf{R}_i\mathbf{r} \tag{1}$$

where $\mathbf{r}$ is the vector within the road plane, that is perpendicular to the driving direction and w_i^{left}, w_i^{right} are the offsets to the left and right ego-lane borders. See Fig. 2 (right) for an illustration. We make the simplifying assumption that the road surface is flat perpendicular to the direction of the car motion (but we don't assume that the road is flat generally - if our ego path travels over hills, this is captured in our ego path).

Given a frame i, we can project all future lane borders $\mathbf{b}_j$ ($\mathbf{b}_j \in \{\mathbf{b}_j^{left}, \mathbf{b}_j^{right}\}$ and $j > i$) into the local pixel coordinate system via

$$\hat{\mathbf{b}_j} = P\left(\mathbf{R}_i^{-1}(\mathbf{b}_j - \mathbf{c}_i)\right) \tag{2}$$

where $P()$ is the camera perspective transform obtained via OpenSfM [29], that projects a 3D point in camera coordinates to a 2D pixel location in the image. Then the lane annotations can be drawn as polygons of neighbouring future frames, i.e. with the corner points $\hat{\mathbf{b}}_j^{left}, \hat{\mathbf{b}}_j^{right}, \hat{\mathbf{b}}_{j+1}^{right}, \hat{\mathbf{b}}_{j+1}^{left}$. This makes implicitly the assumption that the lane is piece-wise straight and flat between captured images. In the following part, we describe how to get the quantities h, $\mathbf{n}$, $\mathbf{r}$, w_i^{left} and w_i^{right}. Note that h, $\mathbf{n}$ and $\mathbf{r}$ only need to be estimated once for all sequences with the same camera position.

The camera height above the road h is easy to measure manually. However, in case this cannot be done (e.g. for dash-cam videos downloaded from the web) it is also possible to obtain the height of the camera using the estimated mesh of the road surface obtained from OpenSfM. A rough estimate for h is sufficient, since it is corrected via manual annotation, see the following section.

In order to estimate the road normal $\mathbf{n}$, we use the fact that when the car moves around a turn, the vectors representing it's motion $\mathbf{m}$ will all lie in the road plane, and thus taking the cross product of them will result in the road normal, see Fig. 3. Let $\mathbf{m}_{i,j}$ be the normalised motion vector between frames

i and j, i.e. $\mathbf{m}_{i,j} = \frac{\mathbf{c}_j - \mathbf{c}_i}{\|\mathbf{c}_j - \mathbf{c}_i\|}$. The estimated road normal at frame i (in camera coordinates) is $\mathbf{n}_i = \mathbf{R}_i^{-1}(\mathbf{m}_{i-1,i} \otimes \mathbf{m}_{i,i+1})$, where $\otimes$ denotes the cross-product (see Fig. 3). The quality of this estimate depends highly on the degree of our previous assumptions being correct. To get a more reliable estimate, we average all $\mathbf{n}_i$ across the journey, and weight them implicitly by the magnitude of the cross product:

$$\mathbf{n} = \frac{1}{\sum_{i=2}^{N-2} \|\mathbf{n}_i\|} \sum_{i=2}^{N-2} \mathbf{n}_i \tag{3}$$

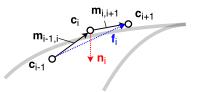We can only estimate the normal during turns, and thus this weighting scheme

Fig. 3. Estimation of the road normal $\mathbf{n}_i$ and forward direction $\mathbf{f}_i$ at a single frame i. The final estimate is an aggregate over all frames.

emphasises tight turns and ignores straight parts of the journey. $\mathbf{r}$ is perpendicular to the forward direction $\mathbf{f}$ and within the road plane, thus

$$\mathbf{r} = \mathbf{f} \otimes \mathbf{n} \tag{4}$$

The only quantity left is $\mathbf{f}$, which can be derived by using the fact that $\mathbf{m}_{i-1,i+1}$ is approximately parallel to the tangent at $\mathbf{c}_i$, if the rate of turn is low. Thus we can estimate the forward point at frame i via $\mathbf{f}_i = \mathbf{R}_i^{-1}\mathbf{m}_{i-1,i+1}$, see Fig. 3. As for the normal, we average all $\mathbf{f}_i$ over the journey to get a more reliable estimate:

$$\mathbf{f} = \frac{1}{\sum_i a_i} \sum_{i=2}^{N-2} a_i \mathbf{f}_i \tag{5}$$

$$a_i = \max(\mathbf{m}_{i-1,i}^\top \mathbf{m}_{i,i+1}, 0) \tag{6}$$

In this case, we weight the movements according the inner product a_i in order to up-weight parts with a low rate of turn, while the max assures forward movement.

w_i^{left} and w_i^{right} are crucial quantities to get the correct alignment of the annotated lane borders with the visible boundary, however automatic detection is non-trivial. Therefore we assume initially that the ego-lane has a fixed width w and the car has travelled exactly in the centre, i.e. $w_i^{left} = \frac{1}{2}w$ and $w_i^{right} = -\frac{1}{2}w$ are both constant for all frames. Later (see the following section), we relax this assumption and get an improved estimate through manual annotation.

In practice, we select a sequence with a lot of turns within the road plane to estimate $\mathbf{n}$ and a straight sequence to estimate $\mathbf{f}$. Then the same values

are re-used for all sequences with the same static camera position. We only annotate the first part of the sequence, up until 100m from the end. We do this to avoid partial annotations on the final frames of a sequence which result from too few lane border points remaining ahead of a given frame. A summary of the automated ego-lane annotation procedure is provided in Algorithm 1 and a visualisation of the automated border point estimation is shown in Fig. 4 (in blue).

Algorithm 1 Automated ego-lane estimation

1: Measure height of the camera above road h
2: Apply OpenSFM to get $\mathbf{c}_i, \mathbf{R}_i$
3: Estimate road normal $\mathbf{n}$ according Eq. (3)
4: Estimate forward direction $\mathbf{f}$ according Eq. (5)
5: Derive vector across road $\mathbf{r}$ according Eq. (4)
6: Set $w_i^{left} = \frac{1}{2}w$ and $w_i^{right} = -\frac{1}{2}w$, where w is the default lane width
7: Derive border points $\mathbf{b}_i^{left}, \mathbf{b}_i^{right}$ according Eq. (1)
8: **for** each frame i **do**
9:     Get all future border points $\hat{\mathbf{b}}_j^{left}, \hat{\mathbf{b}}_j^{right}, j > i$ according Eq. (2)
10:     Draw polygons with edges $\hat{\mathbf{b}}_j^{left}, \hat{\mathbf{b}}_j^{right}, \hat{\mathbf{b}}_{j+1}^{right}, \hat{\mathbf{b}}_{j+1}^{left}$

3.2   Manual corrections and additional annotations

Manual annotations serve three goals: (1) exclude erroneous OpenSfM reconstructions (2) to improve the automated estimate for the ego-lane, (3) annotate additional lanes left and right of the ego-lane and (4) annotate non-road areas.

OpenSfM failures happened a few times, but they are easy to spot by the annotator and subsequently excluded from the dataset. In order to improve the ego-lane positions, the annotators are provided with a convenient interface to edit h, w_i^{left} and w_i^{right}. Note that these quantities are only scalars (in contrast to 3D points), and are thus easily adjusted via keyboard input. We provide a live rendered view at a particular frame (see Fig. 5, left), and immediate feedback is provided after changes. Also, it is easy to move forward or backward in the sequence. For improving the ego-lane, the annotators have the options to:

1. Adjust h (applies to the whole sequence)
2. Adjust all w_i^{left} or all w_i^{right} (applies to the whole sequence)
3. Adjust all w_j^{left} or all w_j^{right} from the current frame i on, $j > i$ (applies to all future frames, relative to the current view)

In order to keep the interface complexity low, only one scalar is edited at a time. We observed that during a typical drive, the car is moving parallel to the ego-lane *most of the time*. Also, lanes have a constant width *most of the time*. If both holds, then it is sufficient to use (2) to edit the lane borders for the whole
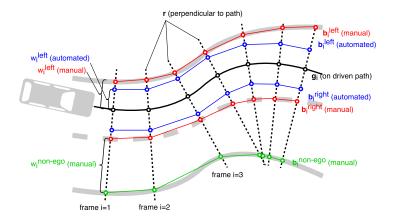
Fig. 4. Visualisation of the lane estimates, seen from above. The automated estimate is shown in blue, while the manual correction is shown in red. A manually annotated additional lane is shown in green. Initially, all w_i^{left} and w_i^{right} are set to a constant value, and thus the estimate is parallel the driven path, which only approximately follows the true lane borders (in blue). Then the annotators can correct w_i^{left} and w_i^{right} for each frame, which moves the border points along **r** (shown as dotted black line) until they align with the true border (shown in red). Furthermore, annotators can add additional (non-ego) lanes and adjust their width $w_i^{non-ego}$.

sequence. Only in the case that the car deviates from the parallel path, or the lane width changes, the annotator needs option (3).

New lanes can be placed adjacent to current ones by a simple button click. This generates a new sequence of $\mathbf{b}_i^{non-ego}$, either on the left or right of the current lanes (see 4). As for the ego-lane, the annotator can adjust the corresponding $w_i^{non-ego}$. Equivalently, a non-road surface can be added next to current lanes, in the same way as if it were a lane, i.e. by getting its own set of $\mathbf{b}_i^{non-ego}$ and $w_i^{non-ego}$. In addition to that, a fixed part on top of the image can be annotated with non-road, as the road is usually found in the lower part of the image (except for very hilly regions or extreme camera angles).

Fig. 5 (left) shows the interface used by the annotators. In the centre of the image, the ego-path can be seen projected into this frame. In the bottom-left, the annotator is provided with controls to manipulate rendered lanes (narrow, widen, move to the left or right, move the boundaries of the lane etc.) and add new lanes. In the top right of the screen (not visible), the annotator is provided with the means to adjust the camera height, to match the reconstruction to the road surface, and the crop height, to exclude the vehicles dash or bonnet. All annotations are performed in the estimated 3D road plane, but immediate feedback is provided via projection in the 2D camera view. The annotator can easily skip forward and backward in the sequence to determine if the labels align with the image, and correct them if needed. An example of a corrected sequence

Fig. 5. Annotator interface with the automated initial ego-lane estimate, given by the future driven path (left) and after manual corrections and additional lane annotations (right). The red text and lines are overlaid descriptions, all other colours are part of the interface.

is shown in Fig. 4 (in red). Fig. 1 shows an example of the rendered annotations and the supplementary material contains an example video.

4 Dataset Statistics and Split

The full annotated set includes 402 sequences, $23,979$ images in total, and thus on average 60 images per sequence. Tab. 3b shows a breakdown of the included annotation types. In total, there were 47,497 lane instances annotated, i.e. 118.2 per sequence. Instance IDs are consistent across a sequence, i.e. consecutive frames will use the same instance ID for the same lane. Furthermore, the annotators have been instructed to categorise each sequence according the scene type: urban, highway or rural. The breakdown of the sequences is shown in Tab. 3a. We plan to update the dataset with new sequences, once they become available.

We split the data into two sets, for training and testing. The train set comprises 360 sequences and a total of $21,355$ frames, while the test set includes 42 sequences and $2,624$ frames. The test set was selected to include the same urban/motorway/rural distribution as the train set. The frames of the training set are made available[1] with both images and annotations while only the images are provided for the testing set.

Furthermore, we have measured the average annotation time per scene type, and find that there is a large variation, with an urban scene taking roughly 3 times longer than a highway or countryside scene of similar length (see Tab. 3). This is due to the varying complexity in terms of the road layout, which is caused by various factors: the frequency of junctions and side roads, overall complexity of lane structure and additional features such as traffic islands and cycle lanes that are typically not found outside of an urban setting.

---

[1] online at https://five.ai/datasets

Table 2. Dataset breakdown according scene type (a) and annotation coverage (b). Coverage of scene types and instances is measured as percentage of the total number of sequences, while the coverage of annotations is measured as percentage of the total number of pixels.

Scene type	
Urban	58.61%
Highway	10.56%
Rural	30.83%

(a)

Annotation type	
annotation density	77.53%
non-road	62.13%
road	15.40%
ego-lane	8.84%
	mean/median/min/max
#instances (per sequence)	2.2/2/1/6

(b)

Table 3. Average annotation time in seconds.

Scene type	Urban	Highway	Rural
Per sequence	361	100	140
Per image	5	2	2

Table 4. Agreement of the annotators

Task	IoU	std
Road vs non-road	97.2 ± 1.5	
Ego vs road vs non-road	94.3 ± 3.4	
	AP@50	AP
Lane instance segmentation	99.0	84.4

The annotation quality is measured through agreement between the two annotators on 12 randomly selected sequences. 84.3% of the pixels have been given a label by at least 1 annotator, with 67.3% of these being given an annotation by both annotators; i.e. 56.8% of all pixels were given an annotation by both annotators. We measure the agreement on these overlapping labels via Intersection-over-Union (IoU) and agreement of instances using Average Precision (AP) and AP@50 (average precision with instance IoU greater than 50%). The results are shown in Tab. 4. The standard deviation is calculated over the 12 sequences.

5   Experiments

To demonstrate the results achievable using our annotations we present evaluation procedures, models and results for two example tasks: semantic segmentation of the road and ego-lane, as well as lane instance segmentation.

5.1 Road and Ego-Lane Segmentation

The labels and data described in 3.2 directly allow for two segmentation tasks: Road/Non-Road detection (ROAD) and Ego/Non-Ego/Non-Road lane detection (EGO). For our baseline we used the well studied SegNet [30] architecture, trained independently for both the EGO and ROAD experiments. In addition to an evaluation on our data, we provide ROAD and EGO cross-database results for CityScapes (fine), Mapillary and KITTI Lanes. We have selected a simple baseline model and thus the overall results are lower than those reported for models tailored to the respective datasets, as can be seen in the leaderboards of CityScapes, Mapillary and KITTI. Thus our results should not be seen as an upper performance limit. Nevertheless, we deem them a good indicator on how models generalise across datasets.

For each dataset, we use 10% of training sequences for validation. During training, we pre-process each input image by resizing it to have a height of 330px and extracting a random crop of 320×320px. We use the ADAM optimiser [31] with a learning rate of 0.001 which we decay to 0.0005 after $25,000$ steps and then to 0.0001 after $50,000$ steps. We trained for $100,000$ training steps, and select the model with the best validation loss. Our mini batch size was 2 and the optimisation was performed on a per pixel cross entropy loss.

We train one separate model per dataset and per task. This leads to 4 models for ROAD, trained on our data, CityScapes (fine), Mapillary and KITTI Lanes. EGO labels are only available for the UM portion of KITTI Lanes and our data, hence we train 2 models for EGO.

For each model we report the IoU, and additionally the F1 score as it is the default for KITTI. We measure each model on held out data from every dataset. For CityScapes and Mapillary the held out sets are their respective pre-defined validation sets, for our dataset the held out set is our test set (as defined in Sec. 3.2). The exception to this scheme is KITTI Lanes which is very small and has no available annotated held out set. Therefore we use the entire set for training the KITTI model, and the same set for the evaluation of other models. We report the average IoU and F1 across classes for each task. Note that we cropped the car hood and ornament from the CityScapes data, since it is not present in other datasets (otherwise the results drop significantly). It should also be noted that the results are not directly comparable to the intended evaluation of CityScapes, Mapillary or KITTI Lanes due to the different treatment of the road occluded by vehicles.

The ROAD results are shown in Tab. 5 and the EGO results in Tab. 6. First, we note that IoU and F1 follow the same trend, while F1 is a bit larger in absolute values. We see a clear trend between the datasets. Firstly, the highest IoUs are achieved when training and testing subsets are from the same data. This points to an overall generalisation issue; no dataset (including our own) achieves the same performance on other data. The model trained on KITTI shows the worst cross-dataset average. This is not surprising, since it is also the smallest set (it contains only 289 images for the ROAD task and 95 images for the EGO task). Cityscapes does better, but there is still a bigger gap to ours

and Mapillary, probably due to lower diversity. Mapillary is similar to ours in size and achieves almost the same performance. The slightly lower results could be due to its different viewpoints, since it contains images taken from non-road perspectives, e.g. side-walks.

Table 5. Results for the ROAD task, measured by IoU and F1 score. Off-diagonal results are from cross-dataset experiments. The column determines which set the model was trained on, and the row determines the source of the evaluation set. The reported column average includes only cross-dataset experiments.

| IoU | Trained On | | | |
	Ours	Mapillary	CityScapes	KITTI
Our Test Set	95.0	85.4	73.2	71.0
Mapillary Val	82.9	90.0	79.6	69.6
CityScapes Val	85.2	85.2	90.0	60.4
KITTI Train	83.8	72.6	74.6	-
Cross-dataset Average	84.0	81.1	75.8	67.0

| F1 | Trained On | | | |
	Ours	Mapillary	CityScapes	KITTI
Our Test Set	97.4	91.9	83.7	81.6
Mapillary Val	90.4	94.7	88.3	81.0
CityScapes Val	91.9	91.9	94.7	74.0
KITTI Train	90.9	83.5	84.8	-
Cross-dataset Average	91.1	89.1	85.6	75.8

Table 6. Results for the EGO task, measured by IoU and F1 score.

Train	Test	IoU	F1
Ours	Ours	88.5	93.7
Ours	KITTI	61.2	72.6
KITTI	Ours	39.2	48.3

Table 7. Results for lane instance segmentation

Metric	Score
AP	0.250
AP@50	0.507

5.2   Lane Instance Segmentation

The annotation of multiple distinct lanes per image, the number of which is variable across images and potentially sequences, naturally suggests an instance segmentation task for our dataset. Though it has been postulated that "Stuff" is uncountable and therefore doesn't have instances [32,33], we present this lane instance segmentation task as a counter example. Indeed it would seem many

stuff-like classes (parking spaces, lanes in a swimming pool, fields in satellite imagery) can have meaningful delineations and therefore instances applied.

Providing a useful baseline for this lane instance segmentation task presents its own challenges. The current state of the art for instance segmentation on Cityscapes is MaskRCNN [34]. This approach is based on the RCNN object detector and is therefore optimised for the detection of compact objects which fit inside broadly non overlapping bounding boxes, traditionally called "Things". In the case of lanes detected in the perspective view, a bounding box for any given lane greatly overlaps neighbouring lanes, making the task potentially challenging for standard bounding boxes. This becomes more apparent when the road undergoes even a slight curve in which case the bounding boxes are almost on top of one another even though the instance pixels are quite disjoint. Recently, a few works have explored an alternative approach to RCNN based algorithms which use pixel embeddings to perform instance segmentation [35,36,37,38]; we provide a baseline for our dataset using pixel embeddings.

Specifically we train a model based on [35]. We follow their approach of learning per pixel embeddings whose value is optimised such that pixels within the same training instance are given similar embeddings, while the mean embedding of separate instances are simultaneously pushed apart. A cost function which learns such pixel embeddings can be written down exactly and is presented in Eq. 1-4 of [35], we use the same hyper parameters reported in that work, and thus use an 8-dimensional embedding space. We impose this loss as an extra output of a ROAD SegNet model trained along side the segmentation task from scratch.

At run time we follow a variant of the approach proposed by [35], predicting an embedding per pixel. We use our prediction of road to filter away pixels which are not likely to be lanes. We then uniformly sample pixels in the road area and cluster their embeddings using the Mean Shift [39] algorithm, identifying the centres of our detected lane instances. Finally, all pixels in the road area are assigned to their closest lane instance embedding using the euclidean distance to the pixel's own embedding; pixels assigned to the same centroid are in the same instance.

For evaluation, we use the Average Precision (AP) measures calculated as described for the MS-COCO [40] instance segmentation task. Specifically: we calculate the AP across images and across IoU thresholds of detected lanes (pixels assigned to embedding cluster centroids) and ground truth lanes. True and false positives are counted in the following way: (1) A detection is a **true positive** when it overlaps a ground truth instance with an IoU above some threshold and (2) a detection is a **false positive** when it does not sufficiently overlap any ground truth instance. Using these definitions we report average precision at 50% IoU and an average AP across multiple thresholds from 50% to 95% in increments of 5%. Tab. 7 shows the instance segmentation baseline results. Qualitatively, the lane instances are well separated, as can be seen in Fig. 6.
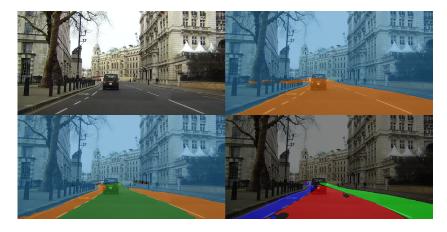
Fig. 6. An example image from our test set (top left) including predictions for the ROAD (top right), EGO (bottom left) and instance (bottom right) tasks. The colours of the ROAD and EGO models match those in Figure 1. The predicted instances are represented by red, green and blue.

6   Conclusions

We have created a dataset for road detection and lane instance segmentation in urban environments, using only un-calibrated low-cost equipment. Moreover, we have done this using an efficient annotation procedure that minimises manual work. The initial experiments presented show promising generalisation results across datasets. Despite this step towards autonomous driving systems, our data has various limitations: (1) Annotations of many other object classes of the static road layout are not included, like buildings, traffic signs and traffic lights. (2) All annotated lanes are parallel to the future driven path, thus currently lane splits and perpendicular lanes (e.g. at junctions) have been excluded. (3) Positions of dynamic objects, like vehicles, pedestrians and cyclists, are not included. In future work, those limitations could be addressed by adding further annotations of different objects in 3D, inspired by [27]. Non-parallel lanes could be handled by extending our annotator tool to allow for variable angles for the lanes in the road plane. Also, a pre-trained segmentation model could be used to better initialise the annotations. Furthermore, the position of dynamic objects could be estimated by including additional sensor modalities, like stereo vision or LIDAR.

Acknowledgements

References

1. Janai, J., Güney, F., Behl, A., Geiger, A.: Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. (2017)
2. Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., Others: An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716 (2015)
3. Oliveira, G.L., Burgard, W., Brox, T.: Efficient Deep Methods for Monocular Road Segmentation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016). (2016)
4. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards End-to-End Lane Detection: an Instance Segmentation Approach. arXiv preprint arXiv:1802.05591 (2018)
5. Brostow, G.J., Fauqueur, J., Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. Pattern Recognition Letters **30**(2) (2009) 88–97
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. CVPR (2016)
7. Neuhold, G., Ollmann, T., Bulò, S.R., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. In: Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy. (2017) 22–29
8. McCall, J.C., Trivedi, M.M.: Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. IEEE transactions on intelligent transportation systems **7**(1) (2006) 20–37
9. Kim, Z.: Robust lane detection and tracking in challenging scenarios. IEEE Transactions on Intelligent Transportation Systems **9**(1) (2008) 16–26
10. Gopalan, R., Hong, T., Shneier, M., Chellappa, R.: A learning approach towards detection and tracking of lane markings. IEEE Transactions on Intelligent Transportation Systems **13**(3) (2012) 1088–1098
11. Li, J., Mei, X., Prokhorov, D., Tao, D.: Deep neural network for structural prediction and lane detection in traffic scene. IEEE transactions on neural networks and learning systems **28**(3) (2017) 690–703
12. Mathibela, B., Newman, P., Posner, I.: Reading the road: road marking classification and interpretation. IEEE Transactions on Intelligent Transportation Systems **16**(4) (2015) 2072–2081
13. Hillel, A.B., Lerner, R., Levi, D., Raz, G.: Recent progress in road and lane detection: a survey. Machine vision and applications **25**(3) (2014) 727–745
14. Brostow, G.J., Shotton, J., Fauqueur, J., Cipolla, R.: Segmentation and recognition using structure from motion point clouds. In: European conference on computer vision, Springer (2008) 44–57
15. Sengupta, S., Sturgess, P., Torr, P.H.S., Others: Automatic dense visual semantic mapping from street-level imagery. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE (2012) 857–862
16. Scharwächter, T., Enzweiler, M., Franke, U., Roth, S.: Efficient multi-cue scene segmentation. In: German Conference on Pattern Recognition, Springer (2013) 435–445
17. Matzen, K., Snavely, N.: NYC3DCars: A Dataset of 3D Vehicles in Geographic Context. In: ICCV, IEEE (2013) 761–768

18. Fritsch, J., Kuehnl, T., Geiger, A.: A new performance measure and evaluation benchmark for road detection algorithms. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), IEEE (2013) 1693–1700
19. Aly, M.: Real time detection of lane markers in urban streets. In: IEEE Intelligent Vehicles Symposium, Proceedings, IEEE (2008) 7–12
20. TuSimple: Lane Detection Challenge (Dataset). http://benchmark.tusimple.ai (2017)
21. Richter, S.R., Vineet, V., Roth, S., Koltun, V.: Playing for data: Ground truth from computer games. In: European Conference on Computer Vision, Springer (2016) 102–118
22. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 3234–3243
23. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In: CVPR. (2016)
24. Leibe, B., Cornelis, N., Cornelis, K., Van Gool, L.: Dynamic 3d scene analysis from a moving vehicle. In: CVPR, IEEE (2007) 1–8
25. Borkar, A., Hayes, M., Smith, M.T.: A novel lane detection system with efficient ground truth generation. IEEE Transactions on Intelligent Transportation Systems **13**(1) (2012) 365–374
26. Laddha, A., Kocamaz, M.K., Navarro-Serment, L.E., Hebert, M.: Map-supervised road detection. In: Intelligent Vehicles Symposium (IV), 2016 IEEE, IEEE (2016) 118–123
27. Xie, J., Kiefel, M., Sun, M.T., Geiger, A.: Semantic instance annotation of street scenes by 3d to 2d label transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 3688–3697
28. Barnes, D., Maddern, W., Posner, I.: Find Your Own Way: Weakly-Supervised Segmentation of Path Proposals for Urban Autonomy. ICRA (2017)
29. Mapillary: OpenSfM (Software). https://github.com/mapillary/OpenSfM (2014)
30. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
31. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)
32. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: ArXiv. (2017)
33. Adelson, E.H.: On seeing stuff: the perception of materials by humans and machines. In Rogowitz, B.E., Pappas, T.N., eds.: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. Volume 4299. (June 2001) 1–12
34. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. CoRR (2017)
35. Brabandere, B.D., Neven, D., Gool, L.V.: Semantic instance segmentation with a discriminative loss function. CoRR (2017)
36. Li, S., Seybold, B., Vorobyov, A., Fathi, A., Huang, Q., Kuo, C.C.J.: Instance embedding transfer to unsupervised video object segmentation (2018)
37. Fathi, A., Wojna, Z., Rathod, V., Wang, P., Song, H.O., Guadarrama, S., Murphy, K.P.: Semantic instance segmentation via deep metric learning. CoRR (2017)
38. Kong, S., Fowlkes, C.: Recurrent pixel embedding for instance grouping (2017)
39. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5) (May 2002) 603–619

40. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755