2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification

Karsten Behrendt*
Automated Driving Team,
Robert Bosch LLC,
Palo Alto, CA 94304
karsten.behrendt@us.bosch.com

Libor Novak*†
Department of Cybernetics,
Faculty of Electrical Engineering,
Czech Technical University in Prague
libornovax@gmail.com

Rami Botros†
ramibotros@gmail.com

Abstract— **Reliable traffic light detection and classification is crucial for automated driving in urban environments. Currently, there are no systems that can reliably perceive traffic lights in real-time, without map-based information, and in sufficient distances needed for smooth urban driving. We propose a complete system consisting of a traffic light detector, tracker, and classifier based on deep learning, stereo vision, and vehicle odometry which perceives traffic lights in real-time.**

Within the scope of this work, we present three major contributions. The first is an accurately labeled traffic light dataset of 5000 images for training and a video sequence of 8334 frames for evaluation. The dataset is published as the *Bosch Small Traffic Lights Dataset* and uses our results as baseline. It is currently the largest publicly available labeled traffic light dataset and includes labels down to the size of only 1 pixel in width.

The second contribution is a traffic light detector which runs at 10 frames per second on 1280×720 images. When selecting the confidence threshold that yields equal error rate, we are able to detect traffic lights as small as 4 pixels in width. The third contribution is a traffic light tracker which uses stereo vision and vehicle odometry to compute the motion estimate of traffic lights and a neural network to correct the aforementioned motion estimate.

I. INTRODUCTION

Automated driving on highways is an actively researched problem which has led to the emergence of many driver assistance systems. Urban areas provide a new set of challenges which require more sophisticated algorithms in multiple areas ranging from perception over behavioral planning to collision avoidance systems. One crucial part of perception is the detection and classification of traffic signs and traffic lights. Traffic lights present a challenging problem due to their small size and high ambiguity with other objects present in the urban environment, such as lamps, decorations, and reflections.

Previous works on traffic light detection and classification utilize spotlight detection and color thresholding [1]–[5], template matching [1], [2], [6], or map information [7], [8]. All these systems make strong assumptions. Usually, they require the traffic lights to be at least a certain size for the algorithm to work [1], [2], on a distinctive background such as suspended traffic lights in front of the sky [4], [9], or assume the existence of maps that contain prior knowledge about the locations of all traffic lights in the environment.

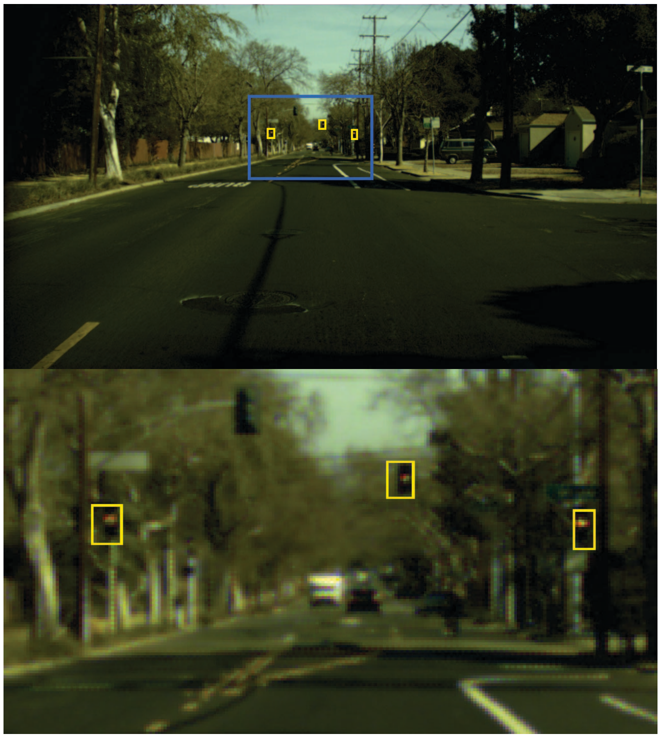

Fig. 1: Sample detections of small traffic lights in an image. The top image is taken at the full resolution of 1280 × 720. At the bottom, the enlarged crop shows detected traffic lights of size about 6 × 12 pixels. All lights are correctly classified as yellow.

With the recent advances and performance of deep neural networks [10]–[13], significant improvements were made in several fields of machine learning and especially computer vision. Deep learning has been used for image classification [10], end-to-end object detection [11], pixel-precise object segmentation [13], and other applications. A neural network approach that creates a probability map for 4 × 4 pixels regions was presented by [15]. Based on the output's probabilities, bounding boxes are fitted to detection clusters. They report a recall of 91.4% with an intersection over union of 0.25 for traffic lights larger than 8 pixels in width.

The drawback of deep neural networks currently is the

*Authors contributed equally
†Authors interned in the Bosch Automated Driving Team at the time of writing.

978-1-5090-4633-1/17/$31.00 ©2017 IEEE

amount of needed training data. Even though there exist datasets for traffic light detection, we present a new dataset, called the *Bosch Small Traffic Lights Dataset* as a contribution of this work. The existing Lara [1] dataset for traffic light detection only offers lower resolution images, while the VIVA [14] benchmark's labeling accuracy is insufficient.

We propose a system for detection, tracking, and 3D localization of traffic lights for automated vehicles, which utilizes deep learning. The detection of traffic lights is carried out by an end-to-end trained neural network [11], which we adapt to detect traffic lights as small as 3 × 10 pixels. The tracker takes advantage of an autonomous vehicle's onboard sensing, as it uses stereo imagery to triangulate the traffic lights' positions in the 3D world and odometry information to estimate the traffic lights relative movement with respect to the vehicle. This location estimate is then corrected by a neural network which is trained specifically for that purpose.

The paper is organized as follows. First, we introduce our dataset in Section II-A and our complete system, starting with the detector in Section II-B. We then give a description of our classifier in Section II-C and tracker in Section II-D. This description is followed by an evaluation and results in Section III.

II. TRAFFIC LIGHT PERCEPTION SYSTEM

Our traffic light detection process consists of four steps as shown in Fig. 2. The first is a bounding box detector which finds traffic lights in images. After that, a classifier removes false positives and predicts the lights' states. Detected traffic lights are then tracked over multiple frames and in the end verified by the classifier once again. In the following, we will go into each of those steps.

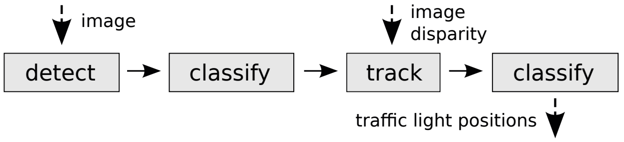

Fig. 2: The detector-tracker-classifier pipeline. The classifier runs twice in order to filter out drifts of the tracker and additional false positives.

A. Dataset

Machine learning approaches, especially deep learning approaches, need data to train on. As part of this work, we publish the *Bosch Small Traffic Lights Dataset* (`http://k0b.de/bstld`), an accurately labeled dataset for detecting, classifying, and tracking traffic lights. Our dataset contains RGB color images at the resolution of 1280 × 720 pixels (see Fig. 7). For training, we collected more than 5000 images, mainly along El Camino Real in the San Francisco Bay Area in California. An overview of the different bounding box sizes is given in Table I, samples are displayed in the Appendix. Overall, 10,756 traffic lights are labeled within those images. The different traffic light sizes within the training set vary between approximately 1 and 85 pixels in width with the mean of 11.3 pixels.

	minimum	average	median	maximum
width	1.12	11.18	8.55	98.0
height	0.25	24.32	18.93	207.0
area	0.28	404.52	158.8	20286.0

TABLE I: Training set bounding box sizes in pixels.

The label distribution of the different traffic light states is heavily skewed towards the three most common types which are "green", "red", and "yellow". Also, because of the difference between the sampling frequency of our camera and the traffic light refresh rate, a lot of traffic lights frequently appear to be off. Whenever this is the case, we opted for labeling them as "off" instead of their current state. The complete distribution of labels within the training set is shown in Table II.

	Training	in %	Test	in %
red	3057	28.42	5321	39.44
red straight	9	0.08	0	0
red straight left	1	0.01	0	0
red left	1092	10.15	0	0
red right	5	0.05	0	0
yellow	444	4.13	154	0.01
green	5207	48.41	7569	56.10
green straight	20	0.19	0	0
green straight left	1	0.01	0	0
green straight right	3	0.03	0	0
green left	178	1.65	0	0
green right	13	0.12	0	0
off	726	6.75	449	0.03

TABLE II: Label distribution over the training and test set.

For testing, we collected a stereo video sequence with odometry along University Avenue in Palo Alto, California. In contrast to the training images, those images are taken and labeled consecutively at a frequency of 15.6 frames per second. Since the bounding boxes are labeled consecutively, it is possible to evaluate a tracker on them.

Overall, 8334 image files are labeled for the test set which contain 13,493 traffic lights. Out of these, 2094 are labeled despite being occluded by a diverse range of objects. Coincidently, our test-set only contains labels of the types "off", "green", "red", and "yellow". Table III gives an overview of the different sizes of traffic lights within our test set. This dataset should be a challenging benchmark for detecting traffic lights or more generally, small objects in images. With this dataset, we provide our results as baseline.

	minimum	average	median	maximum
width	1.875	9.430	8.500	48.375
height	3.250	26.745	24.500	104.500
area	11.718	313.349	212.109	4734.000

TABLE III: Test set traffic light bounding box sizes in pixels.

B. Detection

Detecting objects in images is a challenging task, especially if it is to be done reliably, in real-time, and work for both small and large objects. For traffic lights, real-time detections with low false negative and false positive rates are needed. A high false negative rate results in missed traffic

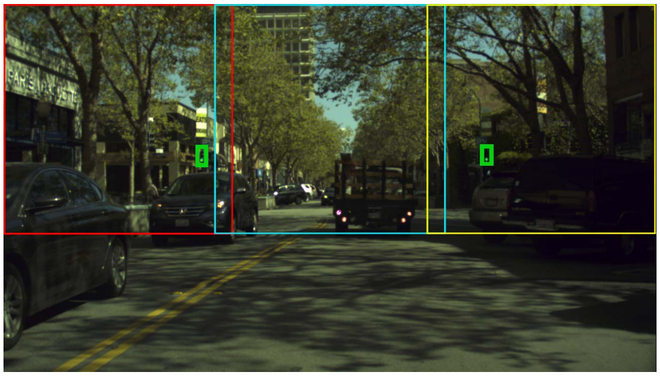

Fig. 3: Sample image from our test-set. Our network is evaluated at the positions within the image, shown in red, teal, and yellow. Two green traffic lights are detected, one by a cell of the yellow network, the other by red.

lights which is not acceptable for autonomous vehicles. False positive detections can lead to the automated vehicle behaving unpredictably e.g., stopping for non-existent red traffic lights.

The "You Only Look Once" (YOLO) [11] architecture shows very good results on PASCAL VOC 2007 and 2012 while processing images at 45 frames per second. One problem with a grid based approach to detection is the limited number of suggested bounding boxes per cell. The authors describe problems detecting small objects and flocks of objects. Another problem is the trade-off between image input sizes, speed, and memory usage. Since the average width of the traffic lights in our datasets is only 10 pixels, we have to adapt the network to detect small objects.

As the first step, instead of taking the complete image as network input, we use the same network to evaluate different patches of the image. During training, random crops of the image are used as input to the network which increased convergence speed and accuracy significantly. For the final detection process, we used three crops in the upper part of the image because most traffic lights are found in that area. We found that by creating a heatmap of traffic light occurrences. Each network has a receptive field of 448×448 pixels. The employed networks' fields of view are visualized for a sample image in Fig. 3.

To achieve a more fine grained detection, we also change the number of grid cells from the suggested 7×7 to 11×11. The granularity of the grid cell was chosen by increasing the number of cells until the network stopped improving. Especially distant traffic lights seemed to be affected by this change.

Initial tests using the original YOLO architecture showed lower performance in detection when the classification part of the network was used. Therefore, since our object classes only differ in the traffic lights' states, we further modify the network model to only detect objects. Removing the classification part from the YOLO architecture yields the following loss function:

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left((x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \right) \\
&+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left((\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \right) \\
&\quad + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(p_i \right)^2 + \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(p_i - \hat{p_i} \right)^2
\end{aligned} \tag{1}$$

For a detailed explanation of all terms, please refer to [11]. However, p_i and $\hat{p_i}$ represent the network's confidence and the intersection over union of the object respectively, within this work. Removing the classification from the detection improved the model's detection results significantly.

The network outputs $11 \cdot 11 \cdot 3 = 363$ bounding boxes with their corresponding confidences per image. Each bounding box is represented by its location (x, y), width, height, and the network's estimate of the bounding box's accuracy. During training, the accuracy estimate is trained with $\hat{p_i}$ being the intersection over union with the detection and the underlying object in the image. As in [11], this part of the loss function is only used if an object overlaps with the prediction. If there is no object overlap, $\hat{p_i}$ is added to the training cost as displayed in (1). After training, most false positives can be removed by simply thresholding based on the networks' own confidence estimates.

The original classification capabilities of the architecture is replaced by a small neural network which is described in the next section.

C. Classification

The states of all detected traffic lights need to be determined. To do so, we create a small classification network that differentiates between the different traffic light states and additionally removes false positives. All bounding boxes are expanded and rescaled so that the traffic lights are 20 px wide and the whole crop is 64×64 px. This provides approximately 22 pixels of context on the left and right. The extra margin gives regional context which is necessary for classification. Without the additional context, for example, traffic light poles or parts of cars (in case of false positives) would not be taken into account.

A dataset for training is created by randomly cropping bounding boxes around the ground truth object with additional image augmentation. The image augmentations include changes in brightness, noise, cropping, stretching, and skewing. Overall we create 75000 crops for training and 4200 for validation. With the limited samples of labeled data, we restrict our classifier training to the "background", "green", "yellow", "red", and "off" classes. The "off" class is necessary for single frame classification since traffic lights often appear as turned off for several frames at a time in our camera images.

The classifier model is shown in Table IV. To ensure classification speed, the model is kept fairly small and can be

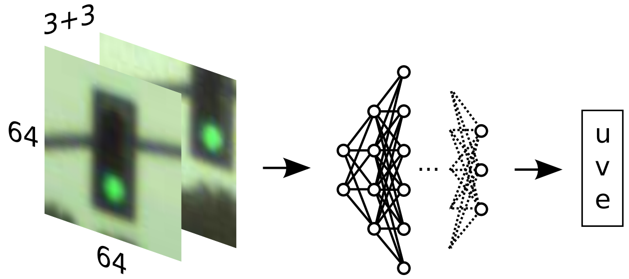

Fig. 4: Neural network for tracking. The neural network receives a prototype image and a crop from a candidate position. It then predicts the candidate's offset (u, v) from the tracked object and an error estimate e.

trained from scratch in only about an hour. Our final model contains 6 weight layers, 3 of which are convolutional and 3 fully connected. All layers use rectified linear units [16] as activation, except for the output layer with a softmax function. In addition, we employ two max-pooling and three dropout layers [17].

Input	3 × 64 × 64
Convolution	filters: 32, kernel: (7,7), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 64, kernel: (3,3), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 128, kernel: (3,3), padding: 0, ReLU
Fully connected	units: 256, ReLU
Dropout	p: 0.5
Fully connected	units: 128, ReLU
Dropout	p: 0.5
Fully connected	units: 5, softmax

TABLE IV: Small classifier model. The network is trained within about an hour and achieves and accuracy of 99.24% on our validation set.

D. Tracking

In order to fill in occasional gaps from the detector we also employ an object tracker. Visual object trackers use a similarity measurement metric on the prototype image to determine its position in a neighborhood around the position in the previous image frame. There are many metrics based on pixel-wise similarity [18], histogram similarity [19], HoGs, feature points, lines, neural network encoders [20], and so on (we refer to [21], [22] for details). Apart from these representations, SVMs, gradient descent, boosting, and other methods are used. We employ a different approach where we use an odometry-based motion model to estimate the movement of the traffic lights and a neural network to improve the tracked positions.

a) Initialization: The tracker is initialized with the bounding boxes from the detector, which are provided asynchronously w.r.t. the camera images. This is caused by the detector's processing time being longer than the time between two consecutive video frames, which presents a challenge for synchronization of the detector output and the tracker. Therefore, processing of new detections is triggered by the arrival of a new video frame. As the received detections are a couple of frames in the past, tracking is re-run from their frame of origin until the current frame and then they are fused with the previously tracked bounding boxes from the tracker.

For each new detection we find the closest previously tracked bounding box by taking the mean distance of their four corners. If the distance is smaller than a threshold, which we chose to be $2w$ (double the width w of the tracked bounding box), we update the tracked detection with the new one by replacing its prototype, position, and size. Otherwise we initialize a new object to be tracked. Because the detector's confidences can be misleading and with the possibility of sudden spikes for false positives (described in [23]), we keep decreasing the confidence of all tracked detections that are not re-detected and raising confidence of the ones that are.

b) Motion model: Our motion model takes advantage of the fact that traffic lights are static objects in the environment. Conveniently, since our system is installed on an autonomous vehicle, which is equipped with stereo cameras and an inertial measurement unit, we can use triangulation and odometry information to create a motion model.

First, our stereo camera system computes a disparity map for each video frame and each traffic light is triangulated into the vehicle reference frame. The median of disparity values in the bounding box is used to represent the entire traffic light. This allows us to better deal with noise in the disparity values. Then, we use linear triangulation [24] to reconstruct the 3D coordinates of the 4 corners of the bounding box $\mathbf{X}_{t-1}^* = [x^c, y^c, z^c]^T$ (c represents a corner id) in the previous vehicle reference frame $t-1$ using the transformation from the camera frame to the vehicle reference frame. Our odometry system gives us the transformation T_{t-1}^t between the vehicle reference frames for the time steps $t-1$ and t. We can then write

$$\bar{\mathbf{X}}_t^c = T_{t-1}^t \bar{\mathbf{X}}_{t-1}^c, \quad (2)$$

$$\bar{\mathbf{x}}_t^c = P \bar{\mathbf{X}}_t^c, \quad (3)$$

where P is the projection matrix from the vehicle reference frame into the camera image frame, $\mathbf{x}_t^c$ are the re-projected image coordinates of the c corner, and $\bar{\cdot}$ represents the homogeneous coordinate notation. A bounding box estimate in the time step t is constructed from the 4 re-projected corners. This bounding box position is refined with a neural network as described below.

c) Position refinement: Tracking traffic lights as small as $3-4$ pixels in width represents a challenging task. Their dark pattern may not yield too many feature points, especially if in front of unlit buildings or if there are trees in the background. In addition to that, traffic lights flicker with a frequency given by the difference between the camera frame rate and the traffic light refresh rate. It is because the Shannon sampling theorem does not hold for the camera and traffic light sampling rates. Also the traffic lights' states may change during the time of tracking.

In order to deal with these conditions, we train a neural network (model shown in Table V), inspired by the spatial

Input	$6 \times 64 \times 64$
Convolution	filters: 16, kernel: (3,3), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 32, kernel: (3,3), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 64, kernel: (3,3), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 32, kernel: (3,3), padding: 0, ReLU
Fully connected	units: 128, ReLU
Fully connected	units: 64, linear
Fully connected	units: 3, linear

TABLE V: The tracker position refinement network.

transformer network [25], which is capable of estimating the misplacement of a traffic light from a prototype image. It takes a prototype and a candidate bounding box estimated by the motion model. They are expanded and rescaled so that the traffic lights are of the reference width 20 px and the whole crop is 64×64 px. All three channels are taken from both images and combined to a $6\times64\times64$ tensor (see Fig. 4). The output is a 3-element vector $[u, v, e]^T$, where u and v are the coordinates of the traffic light in the candidate image and e is the estimated error of the coordinate estimate. The error estimate e represents the uncertainty of the position estimate and is used to trigger the update of the prototype image. We train the network on random crops from our training set with the loss function defined as

$$\text{loss} = \frac{1}{2N}\sum_{i=1}^{N} d_i + \left(e_i - \sqrt{d_i}\right)^2, \tag{4}$$

$$d_i = \left(u_i - u_i'\right)^2 + \left(v_i - v_i'\right)^2, \tag{5}$$

where N is the batch size, $[u_i', v_i']^T$ are the data position labels, which we are learning.

d) Prototype updating: In order to increase the invariance of the tracker to gradual changes of the prototype, we employ prototype updating. We use the error estimate e from the position refining neural network to trigger the update of the prototype. Every time the error estimate e is below a defined threshold, we crop a new prototype image from the current image frame and replace the existing one with it.

The proposed method handles small traffic lights of width $\geq$ 6 px very well and deals reasonably with traffic lights $3-6$ px in width. Additionally, the neural network may be trained to overcome changes in illumination, noise, partial occlusion, and also changing traffic light states. Traffic light state transitions are a big challenge for other appearance-based methods [22] as they change the appearance of the tracked object entirely. Moreover, the fact that our motion model is not dependent on visual features makes it robust to state changes as well.

III. EVALUATION AND RESULTS

Our evaluation is performed on the labeled test-set described in Section II-A. The training and test sets do not overlap in location or even time of recording.

A. Detector

The current implementation of the detector runs at a frequency of 10 fps. Because this is slower than our camera's frame rate, we perform two tests, an offline and an online test. In order to see the maximum potential of the detector, we test it offline which allows it to process each frame. Fig. 5 (top-left) shows the results of this test. It is apparent that the detector is very good in localizing traffic lights which can be seen by its overall high recall values. However, the difference in precision and recall between the intersection over union (IOU) threshold of 0.3 and 0.5 suggests that the detector needs further improvement in bounding box size estimation.

Our system is capable of processing traffic lights in real-time. Yet, since the current implementation of the detector needs 100 ms to process each frame, some frames are skipped and instead handled only by the tracker. Skipped frames obviously lead to a drop in recall if the detector is used online and on its own as shown in Fig. 5 (bottom-left).

B. Classifier

We trained a classifier to remove false positives and decide between different traffic light states ”green”, ”yellow”, ”red”, ”off”, and ”background”. It takes about 0.06 ms to classify 32 samples with our model. With an accuracy of approximately 99% on the validation set we reach 95.1% on slightly translated ground truth data from the test-set. As the largest source of error we identified traffic lights labeled as ”off”, which are often falsely classified as ”background”. The difference in accuracy can have multiple reasons, such as slight differences in the types of traffic lights, background, or even lighting. Training a better classifier can be accomplished with smarter or more aggressive data augmentation and collecting more training data.

C. Tracker

The tracker is tested separately on ground truth bounding boxes in order to evaluate its tracking ability without any influence from the detector. We examine the influence of the initial bounding box width on tracking ability. In a single test we fix the minimum bounding box width w px and a track of length l frames. Bounding boxes smaller than w are filtered out and not used during this test. Ground truth tracks shorter than l are discarded and tracks longer than l are split into tracks of length l. We then test how long the tracker is able to track each track.

Fig. 6 (left) shows the results of the test. As can be seen, our tracker performs well starting with $w \geq 6$ px and is also able to track smaller traffic lights for small l ($1 \leq l \leq 20$). It is observed that all visible traffic lights are successfully tracked for $w \geq 15$ px up to track length $l = 80$. The dataset also includes partially occluded traffic lights, which can be challenging to track.

The influence of running the classifier before and after the tracker is shown in Fig. 6 (right). We would like to point out the ability of the classifier to filter out a half of the false positive detections without a significant drop in recall for track lengths $l < 30$.

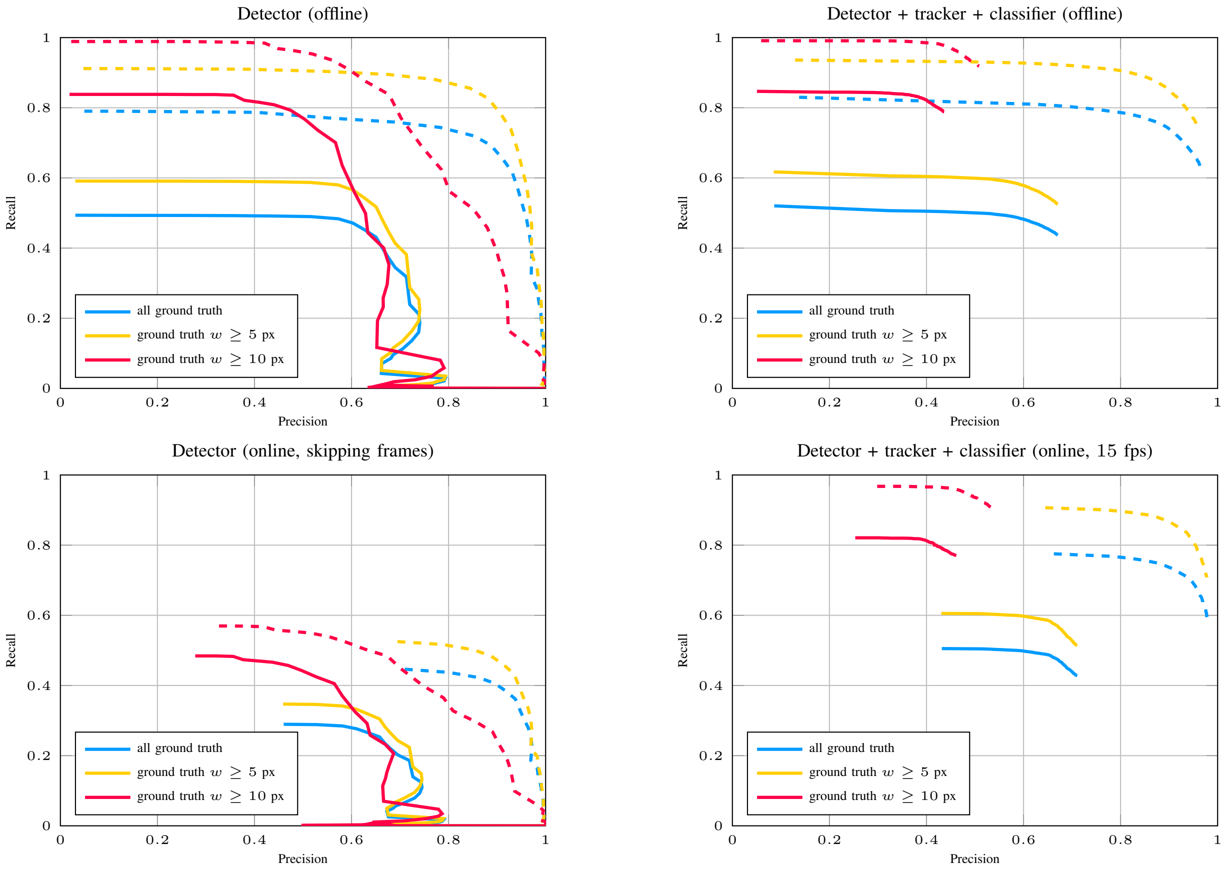

Fig. 5: Precision-recall curves of the system. Solid lines represent an evaluation with intersection over union (IOU) ≥ 0.5, dashed lines an IOU ≥ 0.3. The top row shows an offline run, where each frame is processed by the complete system. The bottom row shows an online run at 15 fps, which causes the detector to skip frames such that the tracker has to fill in for it. The improvement of using the tracker (right) compared to no tracker (left) is clearly visible.

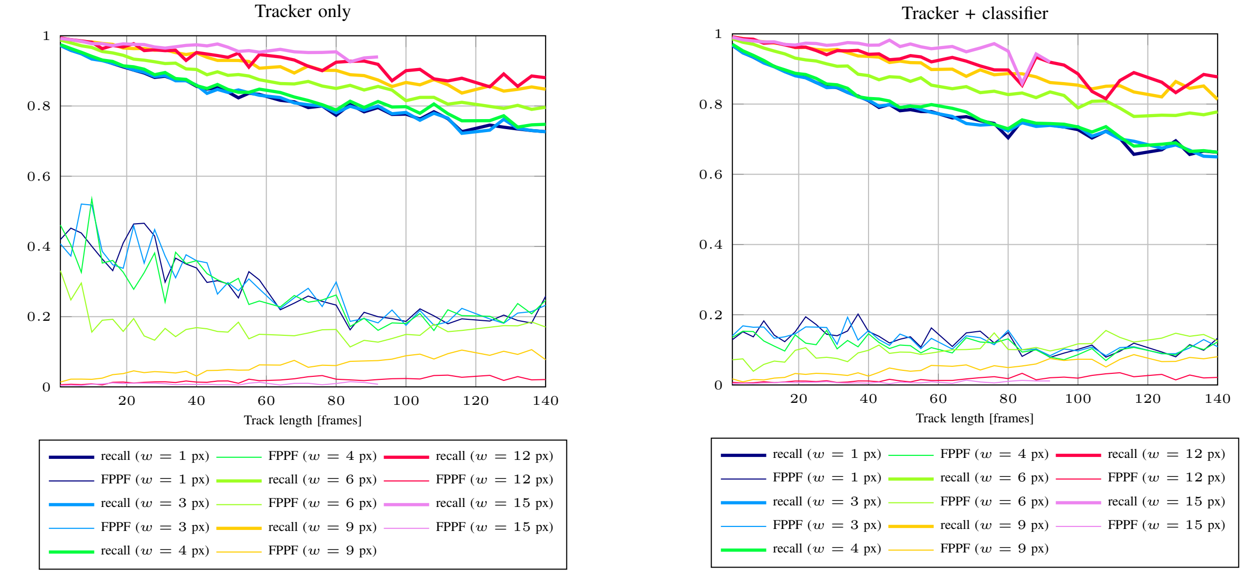

Fig. 6: Tracker evaluation: Testing tracking ability based on initializing bounding box width w and track length. The plots show recall and the false positives per frame (FPPF) of the tracker alone (left) and the tracker together with the classifier (right). Tracks of lengths > 140 frames are not included because they contain, for the most part, static scenes.

D. Detector, tracker, and classifier pipeline

In order to provide an insight on how the whole pipeline improves the results, when compared to the detector alone, we ran an offline and an online test.

a) Offline evaluation: The detections from the offline detector run were used to initialize the tracker. Also, the classifier was run before and after the tracker (as shown in Fig. 2). It is apparent from Fig. 5 (top-right) that the classifier and tracker bring improvements in both precision and recall of the whole system (compared to Fig. 5 (top-left)). As one can notice, the curves are missing the low-recall parts. Since we are plotting precision-recall curves by varying the confidence threshold of the detections, here, we are missing detections with low confidence values. This is caused by our confidence updating mechanism described in Section II-D, which increases the confidence of all bounding boxes that are being re-detected. Therefore, most correct detections will have the maximal confidence value.

b) Online evaluation: Since our system is to be used in autonomous cars, we ran an online test at 15 fps (the frame rate of our camera) to show the real-time capabilities of the whole pipeline. The test was carried out on a computer equipped with an Nvidia GeForce GTX Titan Black and its results are displayed in Fig. 5 (bottom-right). In order to avoid unnecessary overhead, the detections from the detector were filtered by the confidence threshold 0.1. This slightly lowers our recall for this test.

We see that the tracker, combined with the classifier, can substitute the detector on the missed frames extremely well and can push the system to a comparable performance to the offline run (Fig. 5 (top-right)).

Overall, the detection quality of our approach is very high, especially for small traffic lights. For a better visual representation see Fig. 8 or we uploaded a video to `http://k0b.de/tld_icra`. It first shows results of our complete pipeline and then the tracker based on ground truth data, which is being published every few seconds.

IV. CONCLUSIONS

We proposed an approach to detect traffic lights in images at a resolution of 1280×720 pixels using deep learning. First, we created a highly accurate dataset which we are publishing as part of this work under *Bosch Traffic Light Evaluation Dataset*. We trained a neural network for traffic light detection which for equal error rates detects traffic lights down to the size of 4×12 pixels at the IOU of 0.5. Then, a second neural network determines false positive detections and the traffic lights' states. Finally, we describe a stereo vision, odometry, and deep learning based approach to tracking of traffic lights that supplements the detector in case of misses or time constraints. The proposed system runs in real-time and achieves high accuracy in challenging conditions.

As with most machine learning systems, a straightforward approach for improving the accuracy of the system, is to collect and label more training data. Especially, images from other regions would improve the generalization capabilities of our system. Additionally, parallelizing computations in the detector and tracker would allow us to meet even higher real-time requirements.

We believe that our traffic light detection pipeline can lead to smoother trajectories of automated vehicles in urban environments and allow for more accurate mapping systems.

V. ACKNOLEDGMENTS

The authors would like to thank Mithun Jacob from the Automated Driving Team, Robert Bosch LLC for additional supervision of this work.

REFERENCES

[1] R. de Charette and F. Nashashibi, "Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates," in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 358–363.

[2] C. Wang, T. Jin, M. Yang, and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *International Journal of Computational Intelligence Systems*, vol. 4, no. 6, pp. 1383–1390, 2011.

[3] V. Haltakov, J. Mayr, C. Unger, and S. Ilic, "Semantic segmentation based traffic light detection at day and at night," in *German Conference on Pattern Recognition*. Springer, 2015, pp. 446–457.

[4] M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust real-time traffic light detection and distance estimation using a single camera," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3911–3923, 2015.

[5] D. Nienhüser, M. Drescher, and J. M. Zöllner, "Visual state estimation of traffic lights using hidden markov models," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 1705–1710.

[6] G. Siogkas, E. Skodras, and E. Dermatas, "Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information." in *VISAPP (1)*, 2012, pp. 620–627.

[7] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5421–5426.

[8] J. Levinson, J. Askeland, J. Dolson, and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5784–5791.

[9] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended traffic lights detection and distance estimation using color features," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 1315–1320.

[10] 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.

[11] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] 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*, 2015, pp. 3431–3440.

[14] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Traffic light detection: a learning algorithm and evaluations on challenging dataset," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 2341–2345.

[15] M. Weber, P. Wolf, and J. M. Zöllner, "Deeptlr: A single deep convolutional network for detection and classification of traffic lights," in *2016 IEEE Intelligent Vehicles Symposium, IV 2016, Gotenburg, Sweden, June 19-22, 2016*, 2016, pp. 342–348.

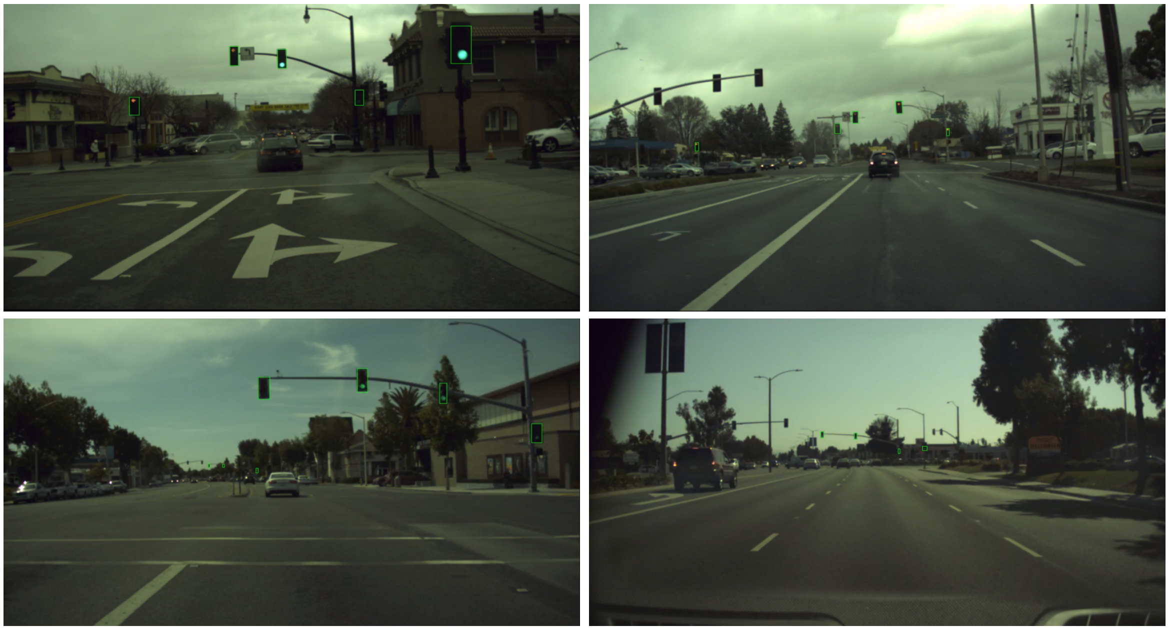

Fig. 7: Training set sample images with annotations. The conditions vary from sunny to light rain.

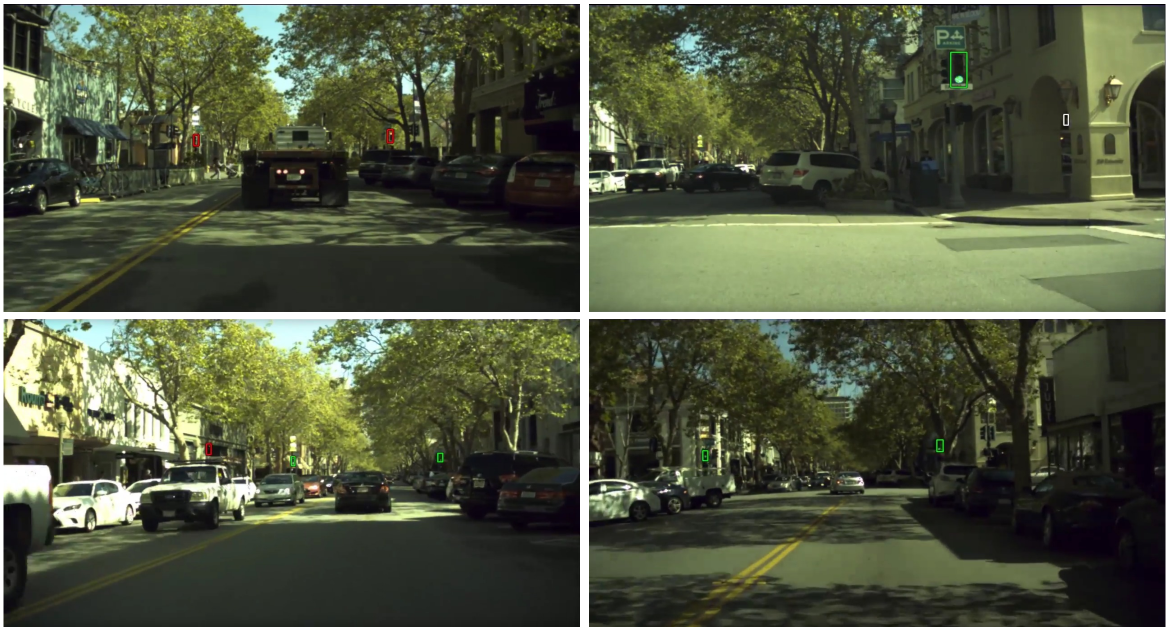

Fig. 8: Sample results. The color of the bounding box represents the classified traffic light state. The system can overcome hard false positives such as brake lights of a truck (top-left), however lamps can still yield false positives (bottom-left).

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[18] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[19] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[20] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *arXiv preprint arXiv:1502.06796*, 2015.

[21] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[22] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

[23] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 427–436.

[24] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.