StixelNet: A Deep Convolutional Network for Obstacle Detection and Road Segmentation

Dan Levi[1]
http://sites.google.com/site/danmlevi
Noa Garnett[1]
noagarnett@gmail.com
Ethan Fetaya[2]
ethan.fetaya@weizmann.ac.il

[1] Advanced Technical Center Israel
General Motors R&D
Herzlyia, Israel

[2] Weizmann Institute of Science
Rehovot , Israel

Abstract

General obstacle detection is a key enabler for obstacle avoidance in mobile robotics and autonomous driving. In this paper we address the task of detecting the closest obstacle in each direction from a driving vehicle. As opposed to existing methods based on 3D sensing we use a single color camera. The main novelty in our approach is the reduction of the task to a column-wise regression problem. The regression is then solved using a deep convolutional neural network (CNN). In addition, we introduce a new loss function based on a semi-discrete representation of the obstacle position probability to train the network. The network is trained using ground truth automatically generated from a laser-scanner point cloud. Using the KITTI dataset, we show that the our monocular-based approach outperforms existing camera-based methods including ones using stereo. We also apply the network on the related task of road segmentation achieving among the best results on the KITTI road segmentation challenge.

1 Introduction

Obstacle detection is a fundamental technological enabler for autonomous driving and vehicle active safety applications. While dense laser scanners are best suitable for the task (e.g. Google's self driving car), camera-based systems, which are significantly less expensive, continue to improve. Stereo-based commercial solutions such as Daimler's "intelligent drive" are good at general obstacle detection while monocular-based systems such as Mobileye's are usually designed to detect specific categories of objects (cars, pedestrians, etc.). The problem of general obstacle detection remains a difficult task for monocular camera based systems. Such systems have clear advantages over stereo-based ones in terms of cost and packaging size.

Another related task commonly performed by camera-based systems is scene labeling, in which a label (e.g. road, car, sidewalk) is assigned to each pixel in the image. As a result full detection and segmentation of all the obstacles and of the road is obtained, but scene labeling is generally a difficult task. Instead, we propose in this paper to solve a more constrained task: detecting in each image column the image contact point (pixel) between the closest

Pages 109.1-109.12
DOI: https://dx.doi.org/10.5244/C.29.109

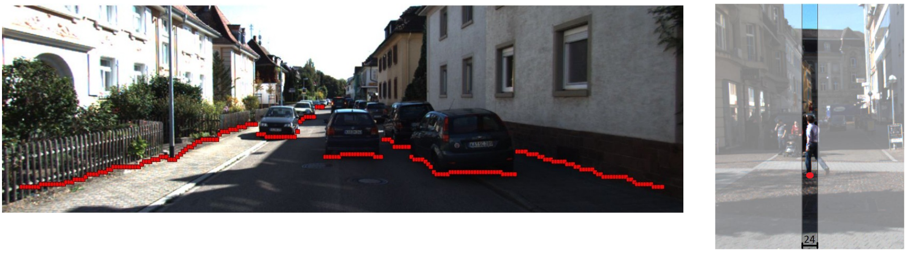

Figure 1: (Left) Obstacle detection example. (Right) Input to StixelNet and output example

obstacle and the ground as depicted in Figure 1(Left). The idea is borrowed from the "Stixel-World" obstacle representation [1] in which the obstacle in each column is represented by a so called "Stixel", and our goal is to find the bottom pixel of each such "Stixel". Note that since we don't consider each non-road object (e.g. sidewalk, grass) as an obstacle, the task of road segmentation is different from obstacle detection as is shown in figure 5(c). Notice also that free-space detection task is ambiguously used in the literature to describe the above mentioned obstacle detection task [1] and the road segmentation task [25].

We propose solving the obstacle detection task using a two stage approach. In the first stage we divide the image into columns and solve the detection as a regression problem using a convolutional neural network, which we call "StixelNet". In the second stage we improve the results using interactions between neighboring columns by imposing smoothness constrains. To train the network we introduce a new loss function based on a semi-discrete representation of the obstacle position probability.

It is well known that having large quantities of labeled data is crucial for training deep CNNs. A major advantage of our unique task formulation is the ability to use laser-scanners, which are excellent at the given task, for labeling, thus eliminating the need for manual annotation. In addition, we further leverage this by fine-tuning StixelNet to the road segmentation task using a smaller amount of hand labeled data. Our experiments use the KITTI dataset [8]. On obstacle detection our approach is the state-of-the-art camera-based method even when compared to the stereo-based "Stixel" approach [1]. On the KITTI road segmentation challenge, our fine-tuned network, although not suitable to model all cases, is ranked second among all methods.

1.1 Related Work

The Stixel representation was presented in [1] as a compact way to represent the 3D road scene and in particularly for obstacle detection. This representation was further developed in several studies [2, 10, 21, 22]. Other methods for obstacle detection include [6, 11]. All of these methods use stereo vision while we use a single camera. A different approach for monocular based obstacle detection relies the host vehicle motion and uses Structure-from-Motion (SfM) from sequences of frames in the video [15, 20]. In contrast our method uses a single image as input and therefore operates also when the host vehicle is stationary. In addition, the SfM approach is orthogonal to ours and can therefore be later combined to improve performance.

For the task of road segmentation, some works such as [24, 25] use stereo data while others such as [17, 19] use a single camera. The work in [19] uses deconvolutional neural networks and is, at this moment, the leading method on the KITTI dataset. These approaches

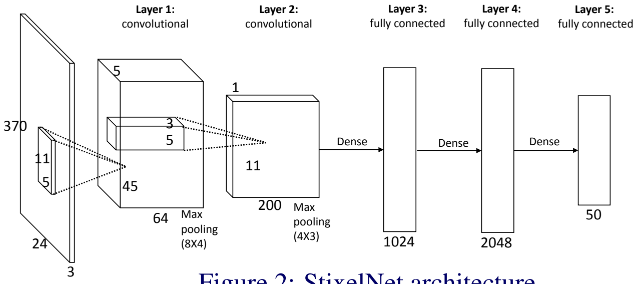

Figure 2: StixelNet architecture.

use pixel or patch level classification as opposed to the proposed column-based regression approach. Our method is novel in providing a unified framework for both the obstacle detection and road-segmentation tasks, and in using the first to facilitate the second in the training phase. The remainder of the paper is organized as follows. In the next section we describe the StixelNet, and in section 3 we describe the post-processing used to obtain the full image obstacle detection and road segmentation. Section 4 describe our experimental results, and in Section 5 we present the conclusions and future work.

2 StixelNet Architecture

The architecture of StixelNet is depicted in figure 2. The network gets as input a single RGB image vertical stripe I_s of dimensions $(w,h,3)$. The problem we are trying to solve is the following: find the pixel location y of the bottom point of the closest obstacle in the center column of I_s. Throughout the paper we will refer to this pixel row value as the "obstacle position" in short. If 0 is the top row of the image and h the bottom row, we consider y to be a real number in the relevant vertical domain $[h_{min},h]$, where h_{min} is the minimum possible row of the horizon given a roughly known camera position. Figure 1(right) shows an example of the input I_s and output y. Accordingly, the output layer represents a probability distribution P over $[h_{min},h]$. $P(y)$ is the probability that the obstacle position in I_s is y. The distribution P can then be used in an image holistic approach to determine the obstacle positions in the entire image, as will be later shown. Experiments with other input types revealed that color images significantly outperform gray ones, while temporal information (e.g. multiple-subsequent frames, optical flow) did not show improvement in our implementation to date.

StixelNet is a multi-layer CNN, inheriting its structure from the famous LeNet [18], in its version implemented in the Caffe framework [14], as the MNIST classification model. It is a 5 layer network in which the first two layers are convolutional and the last three are fully connected. The ReLU non-linearity is applied to the output of all layers except the last one, and for the convolutional layers a Max-pooling layer follows the ReLU operation as well.

In our implementation $w=24, h=370$ and $h_{min}=140$. The first layer convolves the $24\times370\times3$ input image with 64 filters of size $11\times5\times3$ at each pixel position (stride=1). The second layer uses 200 kernels of size $5\times3\times64$. The max-pooling stages compute the maximum over disjoint regions (i.e. there is no overlap between pooled regions), of size 8×4 for the first layer and 4×3 for the second one. The hidden fully connected layers have

1024 ,2048 neurons and the output layer has 50 neurons.

2.1 Loss function

The training examples are given as couples of image stripes and obstacle positions: $(I_s, \hat{y})$. Since this is a regression problem, the natural choice for the networks output is a single neuron whose continuous output is the obstacle position, trained with the L_2 loss function. We refer to this configuration as L_2-*loss*. The drawback of this approach lies in data ambiguities, in which there are several possible choices, for example due to several obstacles, each with some image support. In such cases this network is insufficient in its final layer complexity to model the data.

One solution is to output not only the obstacle position but also a confidence measure. We model this using two continuous network outputs, α, β. The loss function considers these outputs as representing a sigmoid probability function, $P_{Free}(y)\frac{1}{1+e^{-\alpha(y-\beta)}}$. $P_{Free}(y)$ is interpreted as the probability of free space in each row y. Here we define the free space as the area from image bottom to the obstacle position and non-free space the area above the obstacle position. As P_{Free} is defined there is a smooth transition between free space (probability 1) and non free space (probability 0), with β being the most probable transition point and α the confidence level. The label $\hat{y}$ is also transformed to a probability $\hat{P}_{Free}$ as a step function: $\hat{P}_{Free}(y) = 1$ for $y \in [\hat{y}, h]$, and 0 for $y \in [h_{min}, \hat{y}]$). Finally, the loss function, named *KL-loss*, is the Kullback-Leibler divergence between the two probabilities $D_{KL}(\hat{P}_{Free} \| P_{Free})$. We show in our experiments (Section 4) that using the *KL-loss* is significantly better than using the L_2-*loss*. However, the multi-modal nature of the data continues to confuse the network, which in many cases averages several probable y-position outcomes to one improbable result. Returning a few top locations is beneficial since we can pick the right one later in post-processing.

A different approach is to treat the problem as a regular multi-class classification problem. We quantize the output range to N interval segments (bins) with equal lengths, $\frac{(h-h_{min})}{N}$. The label $\hat{y}$ is similarly quantized to N values. The last layer is then composed of N neurons, each representing a bin, trained with the standard *Softmax-loss*. The output of neuron i in the last layer can then be interpreted as the probability of the obstacle position to be in the i^{th} bin. As a single prediction we take the center of the highest probability bin. In the experiments we found that for predicting the correct position the *Softmax-loss* performs better then the continuous versions previously described. The drawback in this representation is the loss of information on the order between the bins and the proximity between neighboring bins. And indeed in practice this loss is less successful at approximating the full probability distribution which is used in post-processing to improve overall results in full images.

As a compromise between the continuous and discrete approaches we introduce a semi-discrete approach, which has shown the best results in our experiments for both predicting the correct obstacle position and for predicting the full probability distribution. In this approach we model the probability $P(y)$ of the obstacle position as a piecewise-linear probability distribution. The network's output layer is a softmax layer of N neurons as in the *Softmax-loss* version. We next define a smooth probability distribution over all the possible values in the output range $[h_{min}, h]$ using linear interpolation between bin centers. Given y between two bin centers: $c_i < y < c_{i+1}$ the probability $P(y) = a_i \cdot \frac{(c_{i+1}-y)}{c_{i+1}-c_i} + a_{i+1} \cdot \frac{(y-ci)}{c_{i+1}-c_i}$, where a_i, a_{i+1} are the output of neurons $i, i+1$ respectively. The loss is defined as the log-probability loss $-\log P(\hat{y})$ and used in the final implementation of StixelNet. We refer to this configuration

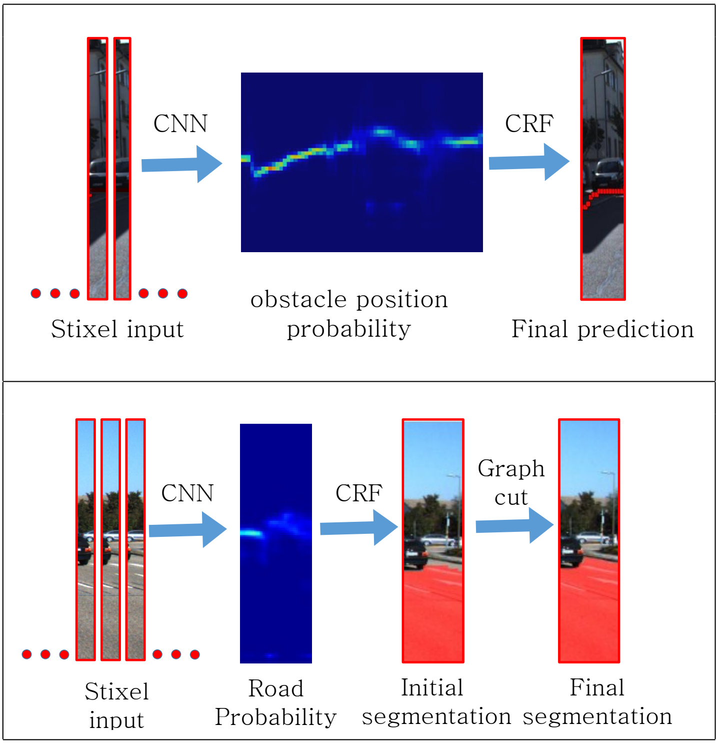

Figure 3: (Top) Obstacle detection algorithm flowchart. (Bottom) Road segmentation algorithm flowchart.

based on the piecewise-linear probability loss in short as *PL-loss*.

2.2 Details of learning

We roughly follow the learning protocol of [16] to train StixelNet, using the Caffe framework [14]. We use stochastic gradient descent with a batch size of 128, momentum 0.9, and no weight decay. The initial learning rate was 0.01 for all experiments except for the ones using the KL-loss which required a smaller learning rate of 0.001 to converge. The learning rate is reduced by half every $10K$ learning iterations. To overcome over-fitting due to our limited training set size we use the dropout technique [12] in the hidden fully connected layers (layers 3 and 4) with the default 0.5 drop ratio parameter.

3 Obstacle detection

We solve the obstacle detection problem in two stages. In the first stage we find a local prediction using StixelNet as described in section 2. In the second stage we use a Conditional Random Field (CRF) in order to get a globally consistent estimation. The flowchart of the obstacle detection algorithm is presented in Figure 3(top).

In the CRF we optimize the following potential: $\mathcal{Y} = \arg\min \sum w_u \phi_u(x_i,y_i) + \sum w_b \phi_b(x_i,y_i,x_{i+1},y_{i+1})$. The unary potential $\phi_u(x,y)$ is the probability that the obstacle position in column x is y as computed by StixelNet. The binary potential between consecutive predictions $\phi_b(x_1,y_1,x_2,y_2)$ is $\min\{\max\{|y_1-y_2|-1,0\},T\}$ where we penalize discontinuities but have zero penalty over small changes. Clipping the potential at T is important to allow large discontinuities at objects boundaries.

The CRF is a chain model over consecutive columns, and therefore the inference can be solved exactly and efficiently using the Viterbi algorithm [23]. Figure 4 shows the wide variety of different scenes in which our obstacle detection operates. Notice the upper leftmost in which green bushes which are upright obstacles are distinguished from the green grass, a result which is difficult to obtain using a scene labeling approach.

3.1 Road segmentation

In road segmentation the challenge is to decide for each pixel if it is road or not. Applying the previous framework, i.e. training StixelNet to find the road limit in each column, and then using a global refinement, has some limitations. The main limitation is that, unlike detection of the closest obstacle, the assumption that the image starts with road at the bottom and then there is a single transition to non-road does not always hold. As an example observe the scene in Figure 5(e) in which the opposite direction road is missed entirely by our method. This is even more evident in junction scenes. Despite these limitations, we will show that we can achieve competitive results with this approach.

The segmentation is done is three stages. The first two, StixelNet (trained on the road segmentation task) followed by a CRF, are the same as in obstacle detection. The final stage performs a graph-cut segmentation on the image to achieve higher accuracy by enforcing road boundaries to coincide with image contours. The flowchart of the road segmentation algorithm is presented in Figure 3(bottom). In the final segmentation the graph structure is a grid over all pixels. The unary potentials are the road/non-road labels from the CRF, and the binary potentials are $\alpha \cdot \exp(-\beta|\nabla I|)$, where α, β are constants. This penalizes road to non-road transitions, unless they coincide with a strong image gradient. Although the resulting graph is loopy, as the potentials are sub-modular, and the task is binary segmentation, it can be solved exactly and efficiently using graph-cuts [7].

4 Experimental Results

For the obstacle detection estimation experiments we use the raw data available in the KITTI dataset [8]. The dataset we chose consists of the on-road sequences, 56 in total, each containing several hundreds of video frames taken from two cameras and synced with additional sensory data. For all our experiments we use a single color rectified image and the Velodyne laser-scanner 3D point cloud for ground truth. For comparison with a stereo-based approach we use the second camera image as well. The images used are rectified to a front view, such that vertical objects in the world appear vertical in the image. We take 6 diverse sequences (09_26_d_27, 09_26_d_35, 09_26_d_64, 09_29_d_71, 09_30_d_27, 10_03_d_47) for testing and the remaining sequences for training. Both in training and testing we use every fifth frame in each sequence. In total there are $\tilde{6}$K training images and 800 testing. We next describe our automatic ground truth method, applied both on the training and testing images.

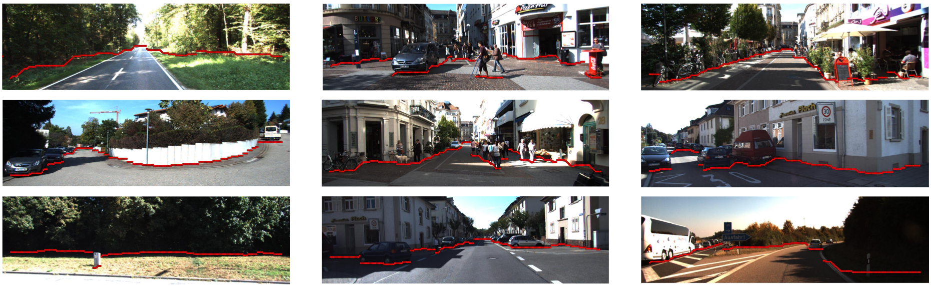

Figure 4: Obstacle detection results on test images (See section 4). the red dots mark the ground contact points of the general obstacles as computed using StixelNet and CRF.

4.1 Automatic obstacle detection ground truth

Given the depth of each pixel in a column, the idea is to find where a vertical obstacle starts, by traversing the column from its bottom looking for an abrupt decrease in depth drop rate. We first create using the Velodyne points a dense inverse-depth map. The Velodyne output is a 3D point cloud which is sparse relative to the image pixels. Each 3D point is projected to a pixel in the image plane using the available calibration between the scanner and the camera, and for each matching pixel (x,y) its depth value $d(x,y)$ is obtained. We then use natural neighbor interpolation [9] to extend the inverse-depth values $(D(x,y) = d^{-1}(x,y))$ to the entire image obtaining a value per pixel. In the next stage we look for the columns in which the $3D$ data is reliable and create a ground truth obstacle position estimation for these columns. Initially, a subset of columns is selected with a stride of 5 pixels on the horizontal axis. For a given column x, we compute the derivative $\frac{\partial D(x,y)}{\partial y}$. We follow the derivative from the bottom, looking for the row in which there is a change from a typical road value to about zero. The row found is marked as the suspect obstacle position and a confidence value is assigned based on the variations between the values of the derivative above and below the point. Finally, several consecutive columns with similar obstacle position are either approved or rejected according their confidence. While somewhat heuristic we found this ground truth procedure highly reliable, although not complete: on average the ground truth covers about a quarter of the columns (See example in figure 5(c)). In total $331K$ ground truth columns are generated from the train set and $57K$ from the test set.

4.2 Obstacle detection

We implement several baseline obstacle detection methods for comparison. First and foremost, we use the stereo-based approach for obstacle detection based on the Stixels representation [1]. This approach uses the disparity map obtained from the stereo cameras to detect the road area, and then finds obstacle position, which is equivalent to the bottom points of the Stixels, by applying dynamic programming to a polar-coordinate occupancy grid. We refer to our implementation of this method as the *Stereo* method in the results below.

Since no previous methods used a single camera for the obstacle detection task as we defined it, we implemented two baselines. A naïve baseline, *MaxGradient*, finds in each column the y-position with the maximal gradient. Our best attempt for a monocular baseline method uses the *HOG+SVM* approach [5]. From the training data, patches are extracted to train a classifier for detecting the obstacle position. The $40 \times 24 \times 3$ color training patches

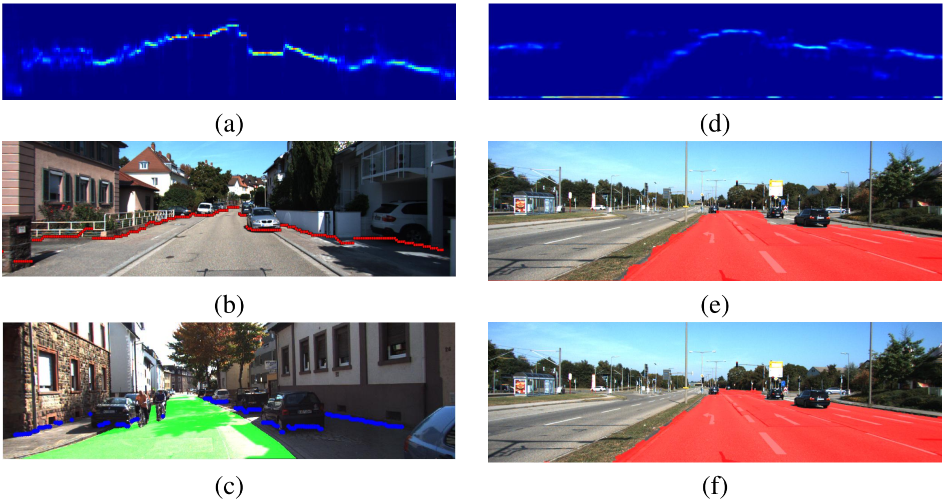

Figure 5: Example results. (a) StixelNet result: obstacle position probability. (b) Obstacle detection (with CRF) overlayed on the image. (c) Obstacle position ground truth obtained with Velodyne laser scanner (blue), and manually labeled road area (green overlay). (d) Road segmentation StixelNet result. (e) Road segmentation (with CRF) (f) Road segmentation (with CRF and graph cut).

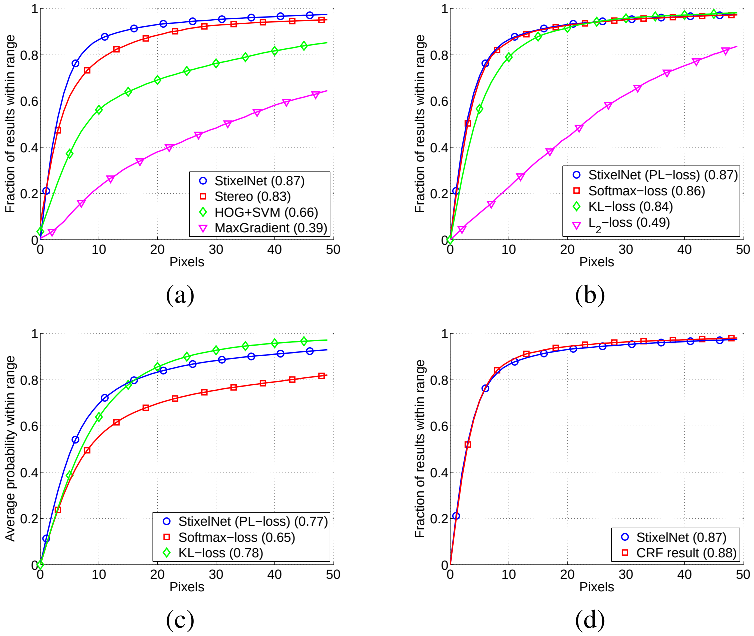

Figure 6: Obstacle detection evaluation (see Section 4 for details). (a) StixelNet vs. baseline methods. (b) Comparison of different losses. (c) Average probability measure comparison of losses. (d) Comparison of StixelNet with and without CRF.

are represented by a HoG descriptor with the default parameters resulting in a 288-length feature vector. Patches centered at the obstacle position are given as positive training examples and sampled patches below it as negative. The SVM is trained in two iterations using hard example mining. Finally, the classifier is applied in a sliding window fashion from the bottom to the top of each column. The first positive local maxima in the classification score encountered is chosen as the obstacle position.

We train StixelNet as previously explained using the generated KITTI obstacle detection training data, running ~ 30 cycles of training. At test time we apply the network on the test image columns with a horizontal stride of 5 pixels. As explained in section 2 the raw output of the network on a particular column is a probability function $P(y)$. As a single candidate for the obstacle position we take $\arg\max P(y)$.

To evaluate the column-based obstacle detection we plot for each value ε of the absolute error in pixels, the fraction of test examples with absolute error below ε. As a single error measure we compute the area under the curve (AUC) for $0 \leq \varepsilon \leq 50$. The evaluation results are shown in figure 6(a). StixelNet with AUC=0.87 outperforms the stereo based-method (AUC=0.82). The median error of StixelNet is less than 3 pixels. The monocular-based baselines, *HOG+SVM* (0.65) and *MaxGradient* (0.33) show relatively poor performance, emphasizing the difficulty of the problem and the superiority of the CNN approach.

Figure 6(b) shows a comparison of the loss schemes described in section 2. StixelNet using the *PL-loss* (AUC=0.87) shows an advantage over the *Softmax-loss* (AUC=0.86) and the *KL-loss* (0.84) across all pixel error values. The L_2*-loss* (AUC=0.51) is clearly not suitable for the problem. The drawback of this evaluation is that it considers a single candidate, while the postprocessing considers the full probability distribution over the possible row positions. To quantify the entire distribution we introduce an additional measure, which is the average probability within ε pixels from the ground truth. Formally, we average over all the test columns $\sum_{|y-\hat{y}|<\varepsilon} P(y)$, where $P(y)$ is the network output and $\hat{y}$ the ground truth label. The results of this evaluation shown in figure 6(c) show a significant advantage of the *PL-loss* compared to the *Softmax-loss*. In addition the *KL-loss* becomes competitive at the right part of the graph due to its ability to avoid large errors.

In figure 5(a) we show the probability distribution output of StixelNet, and below (figure 5(b)) the final result using the CRF. The evaluation in figure 6(d) shows a slight advantage of applying the CRF postprocessing versus using StixelNet only, although the benefits of the CRF in terms of result smoothness and completeness are not reflected this evaluation. The full results on the test videos are provided as supplementary material.

4.3 Road Segmentation

The road segmentation StixelNet is trained and tested on the KITTI dataset. The dataset has 290 train images and 290 test images in which the ground truth road segmentation is manually annotated. The training and testing are divided into three categories: urban unmarked (UU), urban marked two-way road (UM) and urban marked multi-lane road (UMM).

We extracted ~ 44000 stripes at strides of 4 pixels from the original images, and we added the same number taken from their mirror image. We found that over-fitting was still a serious concern, even when using dropout regularization and bagging [4] with 10 nets. In order to get better results, we used a smaller model with 512 neurons in layer 4. The first three layers where fine-tuned from the obstacle detection network. The graph-cut was performed using the code of Boykov *et el.* [3].

The evaluation scheme used for the KITTI road segmentation challenge is described in [13] and includes transforming the image into a "bird-eye view" before performing pixel-wise evaluation of the segmentation. The score used is the F_1 score which is a harmonic mean between precision and recall.

Method	*maxF*	AP	PRE	REC	FPR	FNR
ours (PL+bagging)	89.12 %	81.23 %	85.80 %	92.71 %	8.45 %	7.29%
ours (KL-Loss)	88.48 %	80.96 %	91.06 %	86.05 %	4.65 %	13.95 %
ours (PL-Loss)	88.40 %	77.55 %	86.90 %	89.97 %	7.47 %	10.03 %
DNN [19]	93.43 %	89.67 %	95.09 %	91.82 %	2.61 %	8.18 %
CB	88.97 %	79.69 %	89.50 %	88.44 %	5.71 %	11.56 %
FusedCRF	88.25 %	79.24 %	83.62 %	93.44 %	10.08 %	6.56 %
NNP	87.82 %	76.85 %	86.04 %	89.68 %	8.02 %	10.32 %
ProbBoost [25]	87.78 %	77.30 %	86.59 %	89.01 %	7.60 %	10.99 %

Table 1: The Maximum F1-measure (MaxF), Average Precision (AP), Precision (PRE), Recall (REC), False Positive Rate (FPR) and False Negative Rate (FNR) on the KITTI road segmentation challenge [13].

Table 1 summarizes the results. From the compared methods, FusedCRF uses point clouds from Velodyne and NNP and ProbBoost use stereo data while DNN, CB and our method use monocular vision only. As can be seen, while the StixelNet approach has obvious limitations when applied to road segmentation, it still ranks second best in this challenge.

We also found that there is a difference between the *KL-loss* results and the *PL-loss* results. While overall results are quite similar, the performance on different categories is significantly different. On the UMM task, the *PL-loss* has a maxF score of 93.24% while the *KL-Loss* as a score of 90.3%. On the other hand, the *KL-Loss* outperforms the *PL-loss* with 88.16% vs 84.35% on the UM task.

5 Conclusions

We presented a new column-wise regression approach for obstacle detection and road segmentation from a monocular camera, which uses CNNs, and reaches state-of-the-art results. In the future we plan to increase our method's capabilities by further exploring the interesting possibilities of training with automatically generated ground truth.

References

[1] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3d-world. In *Pattern Recognition*, 2009.

[2] Rodrigo Benenson, Radu Timofte, and Luc J. Van Gool. Stixels estimation without depth map computation. In *ICCV Workshops*, 2011.

[3] Yuri Boykov, Olga Veksler, and Ramin Zabih. Efficient approximate energy minimization via graph cuts. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2001.

[4] Leo Breiman. Bagging predictors. *Machine Learning*, 1996.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[6] Radu Danescu, Florin Oniga, and Sergiu Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Transactions on Intelligent Transportation Systems*, 2011.

[7] B.T. Porteous D.M. Greig and A.H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 1989.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[9] O. Grandstrand. Innovation and intellectual property rights. In *Oxford Handbook of Innovation*. 2004.

[10] Bertan Günyel, Rodrigo Benenson, Radu Timofte, and Luc J. Van Gool. Stixels motion estimation without optical flow computation. In *ECCV*, 2012.

[11] Uwe Franke Hernán Badino and Rudolf Mester. Free space computation using stochastic occupancy grids and dynamic programming. In *In Workshop on Dynamical Vision, ICCV.*, 2007.

[12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[13] T. Kuhnl J. Fritsch and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. *IEEE transaction on intelligent transportation systems*, 2013.

[14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[15] Qifa Ke and T. Kanade. Transforming camera geometry to a virtual downward-looking camera: robust ego-motion estimation and ground-layer detection. In *CVPR*, 2003.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[17] T. Kuehnl, F. Kummert, and J. Fritsch. Spatial ray features for real-time ego-lane extraction. In *Proc. IEEE Intelligent Transportation Systems*, 2012.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[19] Rahul Mohan. Deep deconvolutional networks for scene parsing. *CoRR*.

[20] Jose Molineros, ShinkoY. Cheng, Yuri Owechko, Dan Levi, and Wende Zhang. Monocular rear-view obstacle detection using residual flow. In *ECCV Workshops and Demonstrations*. 2012.

[21] David Pfeiffer and Uwe Franke. Efficient representation of traffic scenes by means of dynamic stixels. In *Intelligent Vehicles Symposium (IV)*, 2010.

[22] David Pfeiffer, Stefan Gehrig, and Nicolai Schneider. Exploiting the power of stereo confidences. In *CVPR*, 2013.

[23] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 1967.

[24] G. B. Vitor, A. C. Victorino, and J. V. Ferreira. Comprehensive performance analysis of road detection algorithms using the common urban kitti-road benchmark. In *Workshop on Benchmarking Road Terrain and Lane Detection Algorithms for In-Vehicle Application on IEEE Intelligent Vehicles Symposium (IV)*, 2014.

[25] G. B. Vitor, A. C. Victorino, and J. V. Ferreira. A probabilistic distribution approach for the classification of urban roads in complex environments. In *Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots on IEEE International Conference on Robotics and Automation (ICRA)*, 2014.